TU WIEN Informatics

Matter and the Web of Things

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Software & Information Engineering

by

Maximilian Schenk
Registration Number 11908528

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.techn. Wolfgang Kastner
Assistance: Univ.Ass. Dipl.-Ing. Jürgen Pannosch, BSc

Vienna, 27th July, 2023

Maximilian Schenk	Wolfgang Kastner

Erklärung zur Verfassung der Arbeit

Maximilian Schenk

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 27. Juli 2023

Maximilian Schenk

Acknowledgements

I am grateful to Univ.Ass Dipl.-Ing. Jürgen Pannosch for his guidance, support, and contributions throughout this scientific thesis. His expertise and meticulous feedback have enhanced the rigor and quality of this research. Jürgen's dedication to fostering a rigorous scientific environment and providing valuable insights and resources have been invaluable assets in completing this thesis. I am fortunate to have had his mentorship and support throughout this scientific endeavor.

Abstract

The evolution of smart home technologies has transformed the way we interact with our living spaces, but the lack of interoperability among various standards, protocols and manufacturers has posed significant challenges. To address this issue, the Matter standard has emerged as a promising solution, aiming to establish a unified and secure ecosystem for smart homes. This thesis focuses on exploring the potential of the Matter standard by conducting a proof of concept involving two development boards, acting as a light bulb and a light switch, where the switch controls the bulb's functionality. The proof of concept involves setting up a Matter network with the two devices and establishing a seamless communication channel using the Matter protocol. The goal is to demonstrate the interoperability and integration capabilities of the Matter standard within a basic smart home scenario. Furthermore, the thesis develops a mapping from Matter to the Web of Things, providing a standardized representation of their capabilities and functionalities. Therefore, a protocol binding that provides Web of Things devices with the ability to communicate with Matter devices, is developed. Additionally, the mapping enhances interoperability, facilitating easy integration of the light bulb and light switch into a Web of Things network.

The findings of this thesis contribute to the existing body of knowledge by showcasing the potential benefits and challenges associated with implementing the Matter standard for controlling smart home devices. By focusing on the specific case of a light bulb and a light switch, the thesis provides practical insights into the feasibility and usability of the Matter standard in everyday smart home scenarios. These insights can inform future advancements in smart home standards, enabling a more unified and streamlined experience for users.

Contents

CHAPTER 1

Introduction

This chapter serves as the gateway to the research, providing a comprehensive and concise overview of the thesis purpose, objectives, and significance. It sets the stage for the reader by outlining the research problem, presenting the context and background information, and clearly stating the research questions. Additionally, it highlights the relevance of the study within the broader academic or practical context, identifies the gaps in existing literature that the research aims to address, and explains the methodology employed to gather and analyze data.

1.1 Motivation

As smart home technologies continue to advance, users are increasingly seeking seamless integration and control of their diverse devices. However, the existence of multiple proprietary standards and protocols has created a fragmented landscape, hindering the interoperability between devices from different manufacturers and impeding the realization of a unified and cohesive smart home experience.

Recognizing the critical need to address these issues the Matter standard stands at the forefront of this endeavor, by establishing a unified framework and protocol, it aims to enable seamless communication and interoperability among smart home devices, irrespective of their brand or underlying technology. Only a Matter certification is required to seamlessly integrate new devices into the Matter ecosystem. This standardization effort holds significant potential for improving interoperability, enhancing the overall functionality, and delivering a more integrated and efficient environment. However, this approach does only work for newly developed devices, or ones getting upgraded to conform to the Matter standard.

Parallel to the Matter initiative, the Web of Things (WoT) also shares the objective of enhancing the smart home landscape. Leveraging existing Web technologies, the WoT

provides standardized metadata and technological building blocks that can further enrich the interoperability and functionality of smart home devices.

However, while the potential for seamless integration between Matter and the WoT is promising, challenges arise when investigating the communication options and bridging the gap between these technologies. Addressing these challenges head-on becomes paramount to harnessing the full potential of both frameworks and paving the way for a future where smart home devices seamlessly interact with each other, providing users with unparalleled convenience, control, and an immersive smart home experience.

1.2 Problem Statement

While Matter promises to simplify the installation, configuration and interoperability of smart home devices, currently there is no way of providing a WoT device with the ability to communicate with a Matter device.

For this communication to work, each Matter device, that is to be controllable via the WoT must have its own Thing Description (TD). For this, a so-called Protocol Binding (PB) is necessary, which specifies how the various WoT concepts within the TDs, such as properties, actions, and events, are represented and accessed over Matter.

Within this TD, each attribute, action and event relevant to the WoT gets derived from the Matter device, providing the WoT with all the necessary information to control the Matter device. However, such a PB currently does not exist for deriving TDs from devices operating on the Matter protocol and has to be developed. Furthermore, Matter's implementation on a real-world smart home system remains, due to its novelty, rather untested.

Many manufacturers offer specialized development kits, able to simulate a certified Matter device, therefore providing a platform for testing the functionality and performance of Matter-enabled devices. However, the interoperability between Matter and the WoT remains widely untouched.

Therefore, the problem statement of this thesis resolves around the specific objective of developing a mapping for Matter devices into the WoT, addressing the challenge of enabling control of Matter devices by a device from the WoT. To accomplish this, the development of a PB becomes necessary. This PB will serve as a crucial component that can be incorporated into a TD, allowing seamless integration of Matter devices into the WoT ecosystem, by providing control over said devices.

Before the development of this mapping can begin, a Matter network has to be set up and put into operation. To achieve this, the thesis aims to demonstrate a simple use case wherein a Matter light bulb is controlled using a Matter light switch, which shall get mapped into TDs afterwards.

1.3 Goal

The goal of this thesis is to evaluate the feasibility of the Matter standard in addressing the interoperability and standardization challenges in the smart home industry, by implementing a Matter-enabled smart home system using the nRF52840 Development Kit, and providing them with a WoT conform TD. Therefore, the thesis aims to achieve the following objectives:

- Provide the reader with a short overview of the current state of the art of smart home technologies, relevant for understanding the position Matter aims to claim.
- Conduct a proof of concept by setting up a Matter network, including a Border Router and some development kits, acting as a Matter light bulb and light switch, verifying the seamless control of the bulb using the switch within the Matter ecosystem.
- Develop a PB for Matter, that can be utilized for mapping Matter devices into the WoT ecosystem.

By accomplishing these goals, this thesis aims to provide insights into the setup and utilization of a Matter network, and provide a PB that can be used to map Matter devices into WoT TDs. Enabling WoT devices to take control over Matter devices.

1.4 Methodology

Beginning with a systematic literature review, which involved conducting a thorough search of academic papers, technical reports, and industry publications, this thesis provides a concise understanding of the existing smart home technologies relevant in the context of Matter, and their features. This literature review serves as a formal methodology, ensuring that relevant information is gathered from reliable sources.

Drawing from the official specifications of Matter and the WoT, the thesis develops a simplified overview of the Matter standard, incorporating formal structures necessary for the development of a fitting PB to be incorporated into a TD. This formal methodology ensures that the PB and mapping of Matter are conducted accurately and following established guidelines.

To validate the feasibility and functionality of the Matter standard, the thesis follows the latest Matter documentation. A proof of concept is conducted by establishing a Matter network wherein two nRF52840 Development Kits [1] are flashed and commissioned to emulate a Matter light bulb and light switch. The conduction of the proof of concept reflects a combination of formal and qualitative methodologies, following structured guidelines while also considering practical aspects of device interaction.

[1] https://www.nordicsemi.com/Products/Development-hardware/nRF52840-DK

Simultaneously, the PB is developed by considering Matter's core functionalities and formats. This formal approach ensures that the PB aligns with the standards set by the Matter standard. Additionally, the thesis applies these formats and conventions to the WoT information model, constructing TDs for the Matter light bulb and light switch. The validation of the constructed TDs using the JSON validation schema of the WoT adds another formal element to the methodology.

CHAPTER 2

State of the Art

With various interpretations of its meaning, the term Smart Home needs clarification. Essentially, a smart home is a residence that uses advanced technology and automation systems to control and monitor various home appliances, devices, and systems, commonly referred to as brown, white, red and gray goods[1]. This technology enables homeowners to manage their homes, making their lives more convenient, secure, and energy-efficient. The devices and appliances within a smart home can communicate with each other by exchanging data and responding to predetermined commands. This allows for customized and intuitive home management, including controlling lighting, heating and cooling, entertainment systems, security cameras, and more. Smart homes typically use a central hub or a mobile app to manage and control these devices, providing homeowners with real-time feedback on their energy usage and system performance. [1]

The rapid expansion of technology has led to a significant increase in the number of electronic devices and systems that are integrated into our daily lives. These technologies are designed to enhance convenience, increase efficiency, and reduce energy consumption, making homes more comfortable and eco-friendly. However, the growing complexity of household electronics has led to new challenges, particularly concerning said targeted power consumption and convenience, as the lack of a standardized approach leads to a variety of incompatible systems, not being able to communicate with each other. In order to address these challenges, manufacturers developed more sophisticated electronic systems and household appliances that are energy-efficient and easy to use. These devices are equipped with advanced sensors, microprocessors, and communication technologies that enable them to interact with other devices and automate various tasks. Despite these advancements, there is currently not one sole standard for smart homes, and proprietary systems are dominating the market. This lack of standardization can make it difficult for consumers to choose between different systems and devices and can limit the

[1] https://www.prologistik.com/en/logistics-lexicon/brown-goods

compatibility between devices from different manufacturers. This can lead to frustration for homeowners, who may find themselves locked into a particular brand or system, with limited ability to customize their home automation setup. [1]

As a result, the industry is working towards developing universal standards, like Matter[1] and protocols that will allow different devices to communicate with each other and enable interoperability, in order to improve the convenience and functionality of smart home systems, making them more user-friendly and accessible to a wider range of consumers.

2.1 Home Automation

With home automation, residents can control their homes remotely, ensuring they may arrive at a comfortable room temperature or well-lit home. Smart devices can be programmed according to the homeowner's preferences, further increasing efficiency and reducing energy waste. The integration of home security systems within automation allows residents to monitor their homes from anywhere, providing peace of mind when they are out of the house. Overall, home automation is transforming the way people interact with their homes, making them more comfortable, efficient, and secure. [1]

2.1.1 Energy Saving

Home automation technology has revolutionized the way we consume energy in our homes, making it easier and more efficient to preserve power. Smart thermostats, for example, can learn the habits of the occupants and adjust the temperature accordingly, eliminating the need for manual adjustments and reducing energy waste. In addition, they can be controlled remotely through mobile apps, allowing homeowners to adjust the temperature when they are away from home. Lighting systems are another area where home automation has made significant progress. With the use of motion sensors and timers, lights can be programmed to turn on only when needed and turn off when the room is unoccupied. Smart bulbs can also be used to adjust the brightness and color temperature based on the time of day or activity, further reducing energy usage. Smart power strips are another tool in power preservation through home automation. These power strips are designed to detect when a device is not in use and turn it off automatically, reducing standby power usage. With the increasing number of electronic devices in the home, smart power strips provide an easy and effective way to reduce energy consumption and save money on energy bills. [2, 3, 4]

A study conducted in 2015 showed that even though smart energy management systems, using HAEMS (Home Automation/Energy Management Systems), still being in their infancy, can significantly reduce a house's cost of operation. By taking various environmental factors like the weather and the effects of solar radiation on the heating and cooling of a building into consideration, paired with real-time prices of utility electricity

[1] https://csa-iot.org/all-solutions/matter/

and the resident's individual comfort settings, up to 50 percent reduction of the cost of operation could be achieved. [5]

Overall, home automation provides homeowners with an array of tools to manage and reduce their energy consumption, leading to cost savings and environmental benefits. As technology continues to advance, the possibilities for energy-efficient automation in the home will continue to advance. [3]

2.1.2 Security Systems

Home automation technology for security has emerged as a rapidly growing industry. The integration of smart technology into security systems has provided an array of benefits, such as the ability to remotely monitor and control security devices, which has made it easier for homeowners to protect their properties. These systems also enable real-time notifications and alerts that can be sent to the homeowner's mobile devices in case of any unusual activity. Furthermore, the use of artificial intelligence in home automation security systems has enabled advanced analytics and predictive capabilities. This means that the system can learn from patterns and behaviors in the home, such as the time of day when people typically arrive or leave, and adjust security settings accordingly. This not only provides a more personalized and effective security solution but also helps to conserve energy and reduce electricity consumption. [6][7]

Another benefit of home automation security is the ability to integrate different devices and systems, creating a holistic security solution. This means that smart locks, cameras, sensors, and alarms can all be interconnected, providing a comprehensive view of the home's security status. For example, a smart lock can be programmed to automatically lock the front door when the security camera detects that the homeowner has left the house.

Overall, the integration of home automation technology into security systems has created a more effective and efficient way of protecting homes. With advanced analytics, predictive capabilities, and the ability to integrate different devices and systems, homeowners can have peace of mind knowing that their homes are secure and protected. [1, 2]

2.1.3 Convenience Improvements

In addition to the convenience of controlling devices and appliances remotely, home automation technology has also led to the development of more advanced features that further enhance convenience. Moreover, home automation technology has enabled the integration of devices and systems that were previously not interconnected. For instance, smart home assistants can be used to control home security systems and door locks, enabling homeowners to manage their security systems remotely with ease. This not only improves convenience but also provides a greater sense of security and peace of mind. Another example of the convenience that home automation technology provides is the

ability to create customized scenes or routines[1]. For instance, a homeowner can create a morning routine that automatically turns on the lights, starts the coffee maker, and adjusts the temperature to a comfortable level. This makes it easier to get ready in the morning and start the day on a positive note. [1]

2.2 Wireless Communication Protocols

Wireless communication protocols play a crucial role in enabling smart home technologies to connect, communicate, and operate seamlessly. These protocols provide the backbone for various devices and systems within a smart home ecosystem. Among the commonly used protocols are Wi-Fi, Bluetooth Low Energy (BLE), ZigBee, and Thread. Each protocol offers unique features and advantages that address different aspects of smart home connectivity, such as Internet connectivity, short-range communication, low-power operation, and mesh networking. These protocols form the foundation for smart home devices to interact with each other, allowing users to control and automate their homes with convenience and efficiency. [8]

2.2.1 Wi-Fi

Wi-Fi, is a widely adopted wireless communication protocol that enables devices to connect to the Internet and communicate with each other within a local network. It operates on the IEEE 802.11 standard and utilizes radio waves to transmit data wirelessly. Wi-Fi has become the backbone of modern smart home technologies, providing high-speed Internet connectivity and facilitating seamless communication between devices. One of the key advantages of Wi-Fi is its wide availability. Most homes and public spaces have Wi-Fi networks installed, allowing smart home devices to easily connect to the Internet and access a vast range of online services. This connectivity enables users to control and monitor their smart devices remotely using smartphone apps or Web interfaces. [9]

Wi-Fi operates in different frequency bands, including 2.4 GHz and 5 GHz. The 2.4 GHz band offers a better range and can penetrate obstacles more effectively but may be prone to interference from other devices. The 5 GHz band provides higher data transfer rates and is less crowded, resulting in faster and more reliable connections, although it has a shorter range. [9]

In terms of data transfer speeds, Wi-Fi has evolved over the years, with each generation providing faster performance. The earlier standards, such as 802.11b/g/n, offer lower speeds ranging from a few megabits per second to a few hundred megabits per second. The more recent standards, such as 802.11ac (Wi-Fi 5) and 802.11ax (Wi-Fi 6), offer significantly higher speeds, reaching several gigabits per second. These faster speeds are beneficial for bandwidth-intensive tasks like streaming high-definition video, online gaming, and large file transfers within the smart home network. [9]

[1] `https://support.apple.com/de-de/HT208940`

Wi-Fi networks are secured using encryption protocols such as WPA2 (Wi-Fi Protected Access 2) and the newer WPA3, which help protect the privacy and integrity of the data transmitted over the network. User authentication methods, such as passwords or more advanced authentication mechanisms like WPA3's Simultaneous Authentication of Equals (SAE), ensure that only authorized devices can connect to the network. [10]

Wi-Fi's versatility extends beyond Internet connectivity. It also allows devices within the same Wi-Fi network to communicate with each other. This capability enables smart home devices to work together, share information, and execute coordinated actions. For example, a Wi-Fi-enabled smart thermostat can receive temperature data from Wi-Fi-connected sensors and adjust the heating or cooling accordingly. [9]

While Wi-Fi offers numerous benefits for smart home technologies, there are a few potential disadvantages to consider:

1. **Interference**: Wi-Fi operates in unlicensed frequency bands, which means that other devices using the same frequency, such as microwaves, cordless phones, or neighboring Wi-Fi networks, can cause interference. This interference can lead to reduced signal quality, slower speeds, or even intermittent connectivity issues. However, selecting less congested Wi-Fi channels and utilizing newer Wi-Fi standards can help mitigate interference problems. [11]

2. **Limited Range**: The range of Wi-Fi signals can be affected by physical obstacles like walls, floors, and distance from the router. As a result, devices located far from the Wi-Fi access point may experience weaker signal strength, leading to slower speeds or dropped connections. Range extenders or mesh Wi-Fi systems can help extend the coverage area and address this limitation. [9]

3. **Power Consumption**: Wi-Fi connectivity can consume significant power, especially in devices with limited battery life, such as sensors or battery-powered smart devices. Constantly maintaining a Wi-Fi connection can contribute to faster battery drain compared to other low-power wireless protocols like ZigBee or BLE. Battery optimization techniques and power management strategies can help mitigate this issue. [12]

4. **Security Concerns**: While Wi-Fi networks can be secured with encryption protocols like WPA2 or WPA3, vulnerabilities or misconfigurations can still expose the network to security threats. Weak or compromised passwords, outdated firmware, or unpatched vulnerabilities can potentially allow unauthorized access to the network or compromise sensitive data. Regularly updating firmware, using strong passwords, and implementing additional security measures can help mitigate these risks. [10]

It is worth noting that many of these disadvantages can be mitigated through proper network planning, configuration, and the use of complementary technologies. Additionally, advancements in Wi-Fi standards and technologies continue to address some of these

limitations, providing improved performance, range, and efficiency for smart home applications.

Overall, Wi-Fi provides a robust and widely available wireless communication protocol for smart home devices. Its high data transfer rates, extensive range, and compatibility with a wide range of devices have made it an integral part of modern smart home ecosystems.

2.2.2 Bluethooth Low Energy

BLE, formerly known as Bluetooth Smart, is a wireless communication protocol designed for low-power, short-range communication between devices, it is based on the IEEE 802.15.1 standard. It is an extension of the classic Bluetooth technology but focuses on energy efficiency, making it suitable for battery-powered smart home devices and other Internet of Things (IoT) applications. BLE operates in the 2.4 GHz ISM (Industrial, Scientific, and Medical) band, similar to Wi-Fi and ZigBee, but it uses shorter data packets and lower transmission power, resulting in reduced power consumption. This low-power operation allows BLE devices to operate on small coin cell batteries or other energy-efficient power sources for extended periods, ranging from months to several years, depending on the device and usage. [13]

BLE is designed for short-range communication, typically up to 100 meters in open space, although the effective range may vary depending on environmental factors and device capabilities. It uses frequency hopping to minimize interference from other devices operating in the same frequency band, enhancing robustness and reliability. One of the key advantages of BLE is its compatibility with a wide range of devices. Most modern smartphones, tablets, and computers are equipped with Bluetooth capabilities, allowing easy pairing and communication with BLE-enabled smart home devices. BLE devices can also operate in a hub-less manner, allowing direct communication between devices without the need for a central hub or Internet connectivity. BLE supports two modes of operation: central and peripheral. In the central mode, a device such as a smartphone or a hub takes the role of a central controller and can connect and communicate with multiple peripheral devices simultaneously. Peripheral devices, on the other hand, are typically smart sensors, actuators, or other low-power devices that transmit data to the central controller or respond to commands. [13]

In terms of data transfer rates, BLE provides lower throughput compared to Wi-Fi or classic Bluetooth. It is designed primarily for transmitting small amounts of data, such as sensor readings or control commands. However, BLE supports efficient data transmission through connection intervals and advanced data compression techniques, ensuring reliable communication while minimizing power consumption. BLE incorporates robust security measures to protect data and ensure device authenticity. Encryption algorithms, pairing mechanisms, and authentication protocols are employed to safeguard the communication between devices. BLE also supports privacy features to prevent tracking and eavesdropping. [13]

Overall, BLE offers a power-efficient and versatile wireless communication protocol for smart home devices. Its low-power operation, wide compatibility, and ability to operate in a hub-less manner make it suitable for a broad range of applications, including smart locks, sensors, wearables, and other battery-powered devices within the smart home ecosystem.

2.2.3 ZigBee

ZigBee is a wireless communication protocol designed for low-power, low-data-rate applications, particularly for smart home automation and Internet of Things (IoT) devices. It operates on the IEEE 802.15.4 standard and employs a mesh networking architecture, allowing devices to form robust and self-healing networks. One of the key features of ZigBee is its low-power operation, enabling devices to operate on small batteries for extended periods, ranging from months to years. This power efficiency makes it suitable for devices that require long battery life, such as sensors and actuators within a smart home environment. [14, 15]

ZigBee networks typically consist of three types of devices: coordinators, routers, and end devices. The coordinator serves as the network controller, initiating and managing the network. Routers extend the network's coverage area by relaying data between devices, while end devices are typically battery-powered devices that communicate with routers or the coordinator. The mesh networking capability of ZigBee allows devices to communicate with each other through multiple paths. If one device fails or is out of range, the network can dynamically reroute the data through other devices, ensuring reliable communication. This self-healing feature enhances the network's robustness and scalability, as devices can be added or removed without disrupting the overall network operation. [14, 15]

ZigBee operates in the 2.4 GHz ISM band, the same as Wi-Fi and BLE, but it uses a different channel and frequency-hopping technique to minimize interference. This allows ZigBee devices to coexist with other wireless technologies in the same environment. ZigBee supports multiple network topologies, including star, tree, and mesh. The star topology consists of a coordinator and end devices directly connected to it. The tree topology extends the range by adding routers, while the mesh topology provides the most extensive coverage by allowing devices to connect to each other and form multiple paths within the network. [15]

In terms of data transfer rates, ZigBee focuses on small data payloads and low-latency communication. It is well-suited for transmitting sensor data, control commands, and status updates within a smart home environment. The data rate typically ranges from 20 to 250 kilobits per second, depending on the ZigBee version and implementation. [15]

ZigBee incorporates robust security measures to protect data and ensure network integrity. It supports encryption, authentication, and access control mechanisms to prevent unauthorized access and maintain the privacy of the transmitted data. These security features are essential for maintaining the integrity of smart home systems. [16]

Overall, ZigBee offers a reliable, low-power, and secure wireless communication protocol for smart home and IoT applications. Its mesh networking capabilities, long battery life, and compatibility with a wide range of devices have made it a popular choice for smart lighting systems, thermostats, door locks, and other home automation devices.

2.2.4 Thread

Thread is a wireless communication protocol designed specifically for smart home applications. It is based on IPv6 networking technology and aims to provide reliable, secure, and scalable connectivity for smart devices. Thread utilizes low-power wireless mesh networking to enable devices to communicate with each other and extend the range of the network. One of the key features of Thread is its IPv6-based networking stack. This allows Thread devices to have unique Internet Protocol (IP) addresses, enabling direct communication with other devices on the network and facilitating seamless integration with Internet protocols. Thread leverages existing Internet infrastructure, making it easier to connect smart home devices to the broader Internet ecosystem. Thread operates on the IEEE 802.15.4 standard, using the 2.4 GHz frequency band. It employs a mesh networking topology, where devices communicate with each other and relay messages to extend the network's coverage. This mesh capability ensures that even if one device becomes unavailable or a path is blocked, the network can dynamically find alternative routes, providing robustness and fault tolerance. [17]

Thread networks are typically composed of three main types of devices: border routers, routers, and end devices. Border Routers serve as the gateway between the Thread network and external IP networks, such as the Internet. Routers enable devices to communicate within the network and participate in relaying messages, while end devices are low-power devices that connect to routers and consume minimal energy. One of the primary benefits of Thread is its energy efficiency. The protocol is designed to minimize power consumption, allowing devices to operate on battery power for extended periods. Thread devices can enter low-power sleep modes when idle, conserving energy and prolonging battery life. [17]

Thread incorporates robust security mechanisms to protect the network and ensure data privacy. It employs encryption, authentication, and authorization techniques to secure communication between devices. Each device within a Thread network is provisioned with unique security credentials, and secure network key updates help prevent unauthorized access and ensure the integrity of the network. [17]

Thread is designed to be interoperable, allowing devices from different manufacturers to seamlessly work together within the same network. This interoperability is achieved through the Thread Group, an industry consortium that develops and promotes the Thread standard. Certification programs ensure that Thread-certified devices adhere to the standard, enabling reliable and consistent communication between devices. [17]

Thread's focus on reliability, security, and scalability makes it well-suited for smart home applications. It is commonly used in devices such as smart thermostats, door locks,

lighting systems, and sensors. The combination of IPv6 networking, mesh topology, and low-power operation makes Thread an attractive choice for building robust and interconnected smart home ecosystems.

CHAPTER 3

The Matter Standard

Version 1.0 of Matter, formally known as Project Connected Home over IP (or CHIP), got released in late September of 2022 and is a new smart home standard that has been developed by a coalition of some of the world's leading technology companies, including Apple[1], Google[2], Amazon[3], and the Connectivity Standards Alliance[4]. The standard aims to address one of the biggest challenges facing the smart home industry: the lack of interoperability between devices from different manufacturers. To achieve this, Matter is built on a robust foundation of open standards, using IP-based connectivity to ensure that devices can communicate with each other regardless of their manufacturer. This is a major departure from previous smart home ecosystems, which relied on proprietary protocols that made it difficult to mix and match devices from different brands. [18]

In addition to its emphasis on interoperability, Matter also prioritizes security and privacy. Devices that are certified as Matter-compatible must meet stringent security standards, including end-to-end encryption and secure key management. This ensures that users can trust that their devices and data are secure from malicious actors. Matter is designed to support communication over a variety of connectivity options, including Wi-Fi, Thread/802.15.4, and Ethernet. This gives users flexibility in how they choose to connect their devices and allows for a wide range of smart home applications, from lighting and climate control to security systems and entertainment. One of the most compelling aspects of Matter is its ease of use. Devices that are certified as Matter-compatible will work seamlessly with each other, regardless of their manufacturer. This means that users can mix and match devices from different brands and still enjoy a seamless smart home experience without the need for complex integrations or separate apps. [18]

[1] https://www.apple.com
[2] https://about.google
[3] https://www.amazon.com
[4] https://csa-iot.org

The purpose of this chapter is to provide a condensed but detailed look at the underlying structure and functionality of Matter, which was derived from the official Matter Specification Version 1.0 [19]. The following chapters will unveil the underlying structure and functionality of Matter's protocol architecture.

3.1 Network Architecture

The objective of Matter is to develop a comprehensive communication protocol for smart home devices, utilizing IPv6 as its foundation. This protocol encompasses the Application Layer, defining its implementation on devices in the form of Clusters (see Section 3.6.1), as well as multiple link layers to ensure seamless interoperability. The architecture of the protocol is organized into distinct layers, offering a clear separation of responsibilities and effective encapsulation of the protocol stack, as seen in Figure 3.1. Matter itself takes place, in the Application Layer where it provides a common framework for developers to create applications that can control and interact with smart devices regardless of their manufacturer or underlying technology. Matter utilizes Transmission Contol Protocol (TCP) or User Datagram Protocol (UDP) (and Bluetooth Transport Protocol (BTP) in some cases) for transmission of messages (see Section 3.4). It establishes a unified language that enables devices to understand and respond to commands and requests from applications. IPv6 is used at the Networking Layer, in order to provide a universal communication protocol for smart home appliances. As Matter supports various connectivity options such as Wi-Fi, Ethernet or Thread/802.15.4, different network topologies are supported.

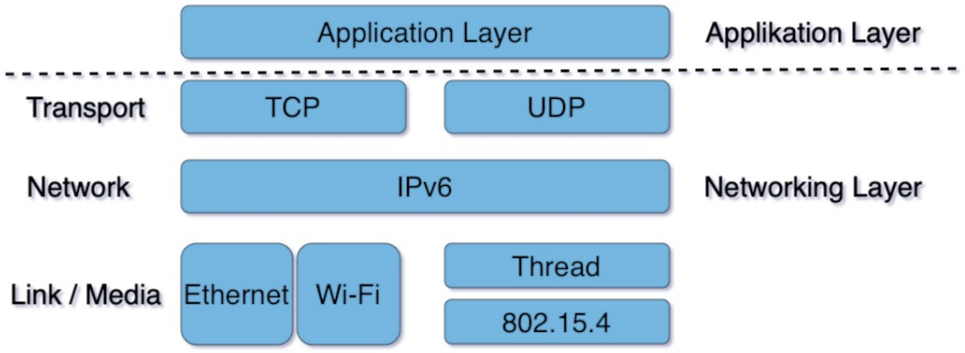


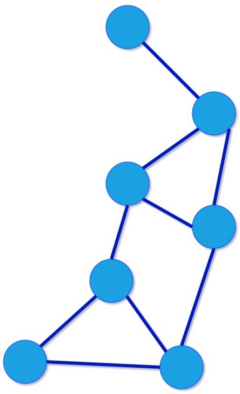
Figure 3.1: Application & Network Stack, adapted from [19]

3.1.1 Network Topology

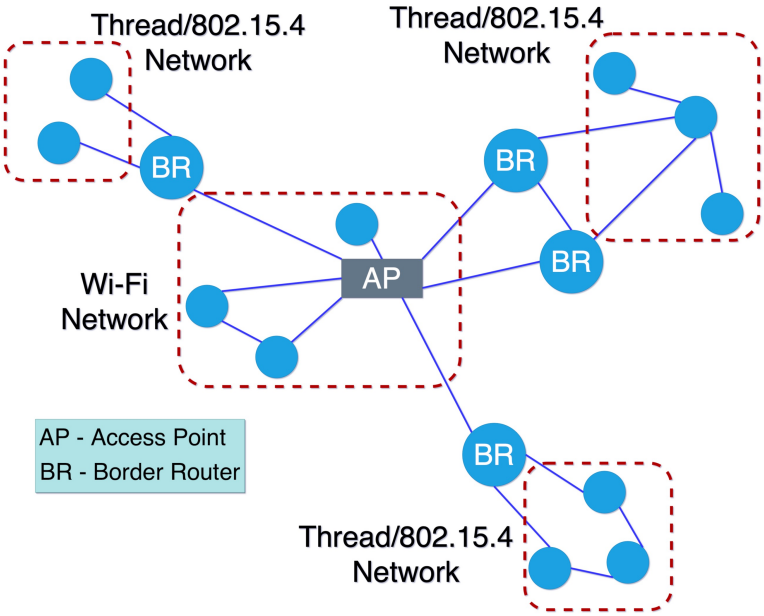
The specification focuses on three specific link layer technologies: Ethernet, Wi-Fi, and Thread/802.15.4. Matter adopts a shared resource approach to networks, allowing the overlay of multiple Matter networks on the same underlying IP networks. This flexibility enables Matter to operate even in network environments disconnected from the global Internet or with limited IPv6 support coming from Internet Service Providers of users.

The protocol also supports local communications across one or more IPv6 subnets through one or more Border Routers. Nodes can have interfaces to a variety of networks, but every communication that crosses the border of a peripheral network (see Figure 3.2b) has to be transferred through a Border Router. The specification deliberately specifies two forms of network topologies supported by Matter: `Single Network` and `Star Network`. Although these designations are reminiscent of typical network terminologies, they do necessarily resemble the same concepts.

In the `Single Network` topology, all Matter devices are connected to one unified network, as seen in Figure 3.2a. Unlike the name could suggest, `Single Network` do not only consist of topologies like `Line` or `Ring` networks but can take on virtually any topology supported by the technology chosen. The term `Single Network` therefore only specifies that the topology does not contain more than one independent network. This network can be established using different technologies like Thread/802.15.4, Wi-Fi, or Ethernet. If Wi-Fi or Ethernet is used, the network can span multiple segments as long as they are bridged together. A Node refers to a single Matter device within the network, and it operates on an IP network. In this setup, each Node can communicate with any other Node in the network using a `Single Network` interface.



(a) Thread/802.15.4 `Single Network`



(b) `Star Network` with multiple peripheral networks

Figure 3.2: Matter Network Topologies

In contrast, the `Star Network` topology functions by connecting numerous peripheral networks to a central hub network, which acts as the central point resembling a star. Typically, this hub network is established as the primary network at the user's home

using Wi-Fi or Ethernet connections. The integration is achieved by directly linking each peripheral network to the hub network through Border Routers. An example of this arrangement can be seen in Figure 3.2b, where a Wi-Fi network serves as the central hub.

In this topology, various peripheral networks can coexist within a single set of nodes called the `Fabric`. The `Fabric` represents a distinct domain within the network where nodes can effectively and securely communicate with each other, abiding by the defined boundaries of Matter.

Nodes within the network possess interfaces that allow them to communicate with both the central hub network and the peripheral networks. This enables direct communication between nodes on the same network. However, when communication needs to traverse network boundaries, it must pass through a Border Router to ensure proper routing and connectivity. Border Routers play a crucial role in the Star Network topology by enforcing specific requirements such as address assignment, route assignment and advertisement within their associated networks.

While the primary supported network protocol in the Matter standard is Thread, the specifications also allow for the possibility of incorporating multiple Thread networks or a combination of different networking technologies to cater to diverse scenarios and enhance interoperability within the smart home ecosystem.

3.2 Layered Architecture

Similar to the well-known Open Systems Interconnection (OSI) Model, the architecture is structured into layers to achieve a clear distinction of responsibilities and establish robust encapsulation among the various components of the protocol stack. This layered design promotes modularity and maintainability throughout the system by organizing functionalities into separate layers. Each layer focuses on specific tasks and maintains well-defined interfaces with neighboring layers, facilitating a modular and scalable approach. By adopting this layered architecture, development, testing, and evolution of the protocol stack become more manageable and efficient, fostering a system that is tailored to meet evolving requirements. Most of the different interactions take place within the stack visualized by Figure 3.3.

The Application Layer handles the high-level business logic of a device, focusing on specific functionalities. In a switch device, for example, it houses the logic that handles controlling the power state of another connected device.

The Data Model Layer encompasses the data structures and verbs that support the application's functionality. It provides a framework for organizing and accessing the relevant data elements. When the Application Layer needs to interact with the device, it utilizes these defined data structures.

The Interaction Model Layer defines a set of interactions between client and server devices. Based on the data elements defined in the Data Model Layer, it specifies how devices can communicate with each other and exchange information.

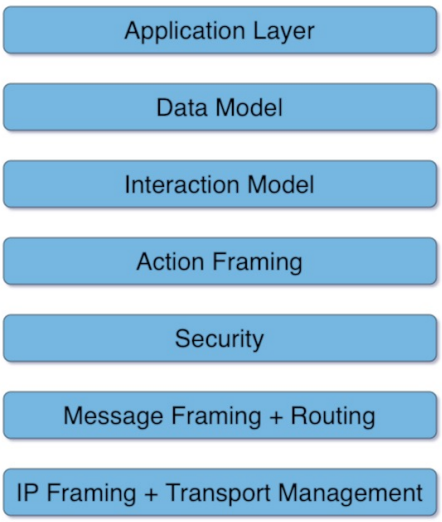


Figure 3.3: Layered Architecture, adapted from [19]

To facilitate network transmission, the Action Framing Layer serializes the constructed interactions into a compact binary format. This format optimizes the size and efficiency of the data for communication across the network.

Ensuring security and integrity, the Security Layer encrypts the serialized interaction and appends a message authentication code to protect the data during transmission, ensuring the confidentiality and authenticity of the communication.

The Message Layer constructs the final payload format, incorporating necessary header fields to convey essential information about the message and its routing. Once the payload is constructed, it is handed over to the underlying transport protocol, such as TCP, UDP or BTP, which manages the transmission of data over the IP network. Upon receipt on the receiving device, the message travels up the protocol stack, with each layer performing reverse operations to interpret and process the data. Finally, the message reaches the Application Layer of the recipient device, where it is consumed and utilized according to the specific requirements and functionalities of the application.

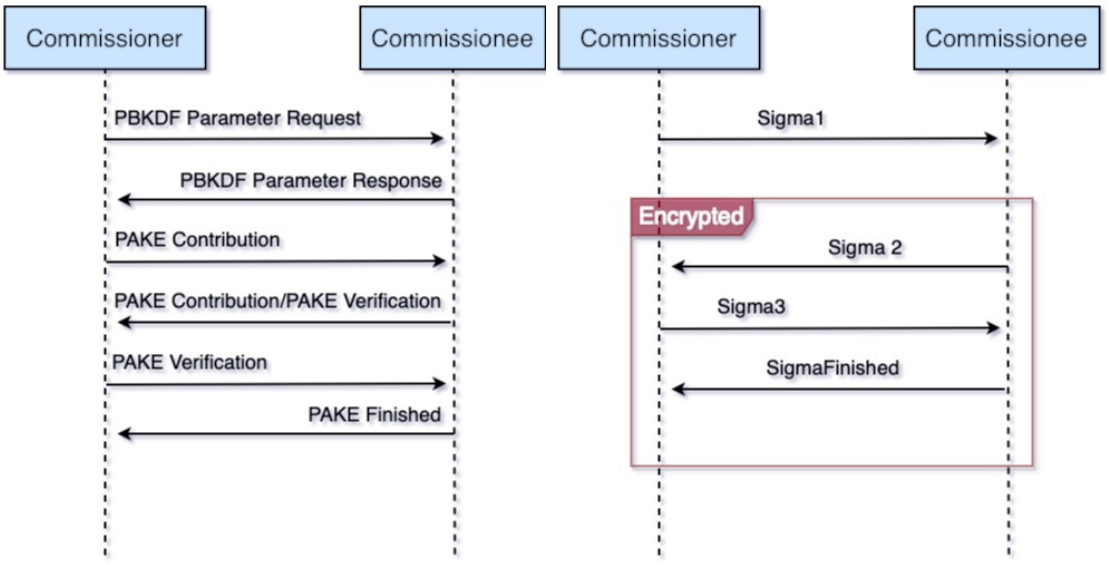
3.3 Session Establishment

Session establishment is a fundamental process in communication systems that allows two or more entities to establish a connection and exchange information in a structured manner. During session establishment, the participating entities negotiate and establish the necessary parameters and protocols to ensure a smooth and reliable data exchange. Matter includes two ways of establishing a session, Passwort-Authenticated Session Establishment (PASE) and Certificate Authenticated Session Establishment (CASE).

In order to provide comprehensive protection to the network, Matter utilizes elliptic

curve cryptography, which is based on the NIST P-256 curve (secp256r1), for public key cryptography and signatures. Shared key cryptographic operations are provided by commonly available AES-128, and may be superseded by AES-256 in the future if Matter's security mechanisms need to evolve.

The PASE, is used in the event of commissioning a new device into an existing Matter network. Its main function is to generate shared keys out of a passcode only one party knows at the beginning. This is accomplished by using a shared passcode in combination with an augmented Password-Based Key Derivation Function (PBKDF) in the form of SPAKE2 [1]. The whole process is based on a PBKDF. A schematic overview of the protocol can be seen in Figure 3.4a.



(a) PASE Protocol Overview (b) CASE Protocol Overview

Figure 3.4: Matter Session Establishment, adapted from [19]

PASE messages are completely unsecured at the Message Layer.

In the first step, the commissioner requests certain PBKDF Parameters from the commissionee, such as a generated random number, a generated session identifier, the Protocol ID and the Protocol Opcode. Upon receipt of the PBKDF Parameter Response, the commissioner sets the Peer Session Identifier to the value of the receipt session identifier. Once this is done, the commissioner generates initiator values used to generate the public key A. When receipt, the commissionee computes the public key B using the passcode acquired earlier and shares it with the commissioner. Both parties derive a shared secret used for the computation of encryption and authentication keys. In the next step, these keys are used for key confirmation. For doing so, the commissionee sends

[1] https://datatracker.ietf.org/doc/pdf/draft-bar-cfrg-spake2plus-02.pdf

a key confirmation message to the commissioner, which then responds with its own key confirmation message. Once both parties have received and validated these messages the protocol is completed and the keys are valid.

The CASE is used to provide exactly two parties with private keys while preserving privacy of both by using Node Operational Credentials. This protocol is derived from the SIGMA [20] protocol. As seen in Figure 3.4b, the protocol is rather short and can be completed within two round trips. At first ephemeral elliptic curve public keys get exchanged between the parties, in order to generate a shared secret. Step two involves exchanging certificates that serve as proof of identity. Finally the possession of a Node Operational Certificate (NOC) and further using it to sign the ephemeral keys and the NOC itself, prove the possession of said NOC.

The strength of CASE lies within its capability of resuming a previous session. Resuming a session excludes computationally expensive steps like signature creation and verification, which makes it an ideal feature for devices demanding low power usage. In order to resume a session, the initiator shares its Resumption ID generated during the Sigma1 challenge (see Figure 3.4b). If both parties remembered the shared secret associated with the Resumption ID, the session can be restored. Once the session is established, devices can start transmitting and receiving secured messages.

3.4 Messages

In Matter, communication is carried out through messages, which can be either secured or unsecured. Each message has specific attributes such as a Session Type and Session ID (both described in Table A.1), which determine whether it is secured and how it should be decrypted and authenticated if it is secured. Additionally, messages have a Message Counter field that serves the purpose of uniquely identifying the message for security and duplicate detection. Matter messages are utilized by both Matter applications and the Matter protocol stack itself to transmit application-specific data or commands.

Messages are equipped with different variables and fields that serve the purpose of tracking interconnected messages involved in small, separate transactions. This message tracking mechanism provides transaction tracking functionality to the Interaction Model Layer, allowing multiple concurrent transactions to be multiplexed over a given session. Furthermore, these variables and fields seamlessly incorporate the Message Reliability Protocol (MRP) as a service, enabling its utilization over UDP transports.

In Matter two types of messages are defined, `Control Messages` and `Data Messages`. They share the same format and only differ in the used implementation of the Message Counter, in order to allow for secure operation on the same security key. Data Messages are only used for internal protocols that initialize security, so most messages in the network are Control Messages.

Matter supports acknowledgment mechanisms to ensure reliable message delivery. When a device receives a message, it may send an acknowledgment back to the sender, confirming

that the message was received successfully. This helps in maintaining the integrity of communication and detecting any transmission errors. However, each message has to follow a specific message format.

The message format in Matter offers versatile support for different communication paradigms, including secure unicast sessions, multicast group messaging, and session establishment. Regardless of the communication mode, the encryption process for Matter messages remains consistent, assuming that communicating parties share symmetric keys. Unencrypted messages are solely utilized for protocols that establish secure messaging, such as session establishments.

Messages follow a specific format and are constructed out of various parameters and components, such as the Message Length, various Flags, security measurements and administrative components, and the Message Payload (for a detailed overview see Section A.1). This format ensures compatibility with a variety of communication paradigms. Regardless of the mode of communication, the encryption process remains the same, with the assumption that symmetric keys are shared between all communication partners. The little-endian byte order is used for the transmission of all multi-byte integer fields.

Each Message Payload has to contain a Protocol Section. This Protocol Section of a message includes a Protocol ID and Protocol Opcode, which identifies the message's semantic meaning and the structure of any associated application payload data (for a detailed overview see Appendix A.2). Furthermore, Matter messages contain an Exchange ID, which associates the message with a specific exchange or conversation between two nodes. Additionally, certain types of Matter messages can include acknowledgments to confirm the receipt of a previous message. This acknowledgment mechanism is employed as part of the MRP, ensuring the reliable delivery of messages over unreliable transport channels, enabling devices to further process important messages.

Each secured Message has to undergo a specific procedure, depending on its origin (incoming or outgoing). For encryption and decryption of messages, the Authenticated Encryption with Associated Data (AEAD) algorithm is used. Its nonce results from the concatenation of the Security Flags, Message Counter and Source Node ID. Depending on the set Operational Node ID of the initiator, the Source Node ID has to be set accordingly. If the Session Type is set to be a Secure Unicast Session, sessions using CASE have to resolve the value through the Secure Session Context in relation to the Session Identifier. PASE sessions on the other hand shall set the value to `Unspecified Node ID`. However, if the Session Type is set to be a Group Session, the S Flag has to be set to 1, and the value must be the Source Node ID of the message, otherwise, if the S Flag is set to 0, the message has to be discarded. Depending if the device is transmitting or receiving a message, it has to undergo specific processes.

3.4.1 Outgoing Messages

Encrypting an outgoing message is an essential part of secure communication. Matter uses AEAD to fulfill this process. At first, the key used for encryption has to be obtained.

This can be done using the affiliated Session ID and Session Type in combination with the Destination Node ID. In case no key can be found, the process fails. The AEAD algorithm gets fed with the obtained key, the nonce generated earlier (see Section 3.4), the Message Payload, and additional data in the form of an octet string (consisting of Message Flags, Session ID, Security Flags, Message Counter, and optionally Source Node ID, Destination Node ID, Message Extensions). Upon success AEAD results in the encrypted output payload, followed by a Message Integrity Check (MIC) of variable length. If the encryption fails, no further security action shall be applied. A schematic of the process can be seen in Figure 3.5.

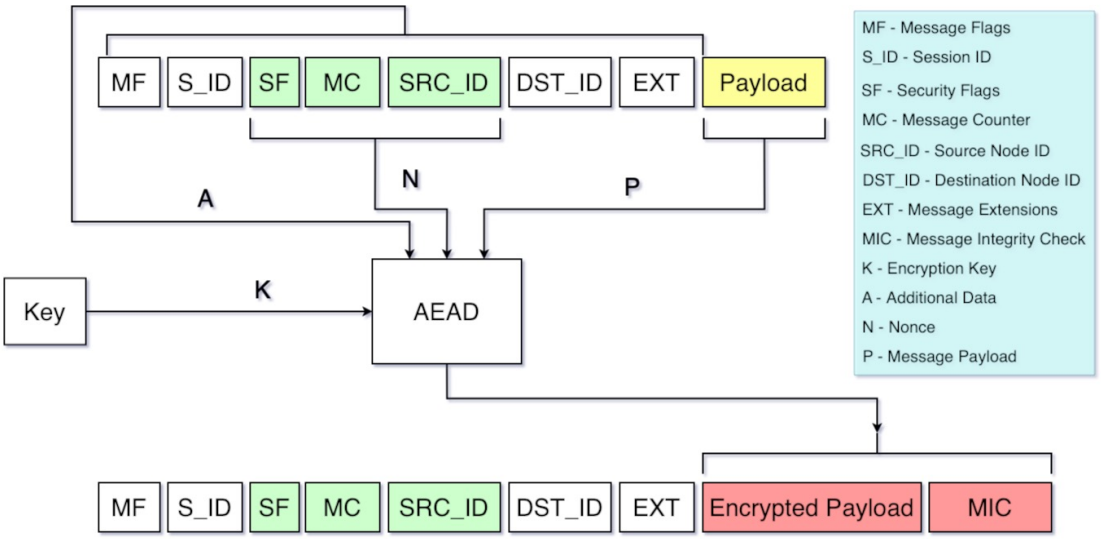


Figure 3.5: Matter Message Encryption, adapted from [19]

3.4.2 Incoming Messages

Similar to the process of encrypting outgoing messages, see Section 3.4.1, the first step of decrypting an incoming message, consists of obtaining the encryption key associated with the given Session ID and Session Type and Message Counter. In case no key can be found, the process fails. The AEAD algorithm gets fed with the obtained key, the nonce generated earlier (see Section 3.4), the encrypted and authenticated Message Payload, and additional data in the form of an octet string. Upon success, the algorithm returns a tuple in the form of `{success, P}` , where `success` represents a boolean value, and `P` is the decrypted Message Payload. If the boolean value results in `FALSE`, the process stops and the upper layer gets notified. Upon `TRUE` the now deciphered and authenticated message gets released to the next processing layer.

3.5 Commissioning

The commissioning process of Matter involves a meticulous and thorough series of steps to set up and configure Matter-compatible devices within a smart home environment, as seen in Figure 3.6. It begins with Device Discovery, where the devices that will be commissioned are identified. This can include a variety of IoT devices. Next, the devices are connected to a suitable IPv6 network infrastructure using Ethernet, Wi-Fi, or Thread as the appropriate link-layer technology.

Once connected, the process continues with Security setup using PASE for establishing secure communication. Device Attestation is then performed to authenticate the identities of the devices and grant necessary permissions. Information Configuration takes place, assigning unique device identifiers, allocating network addresses, and setting up security credentials. Network formation is determined based on the desired topology, whether a `Single Network` or a `Star Network` configuration. Furthermore, Security setup with CASE is carried out to enhance the security of the commissioning process. Device configuration allows for customization of device behavior, such as adjusting lighting preferences or access controls. User-friendly interfaces are provided for easy management.

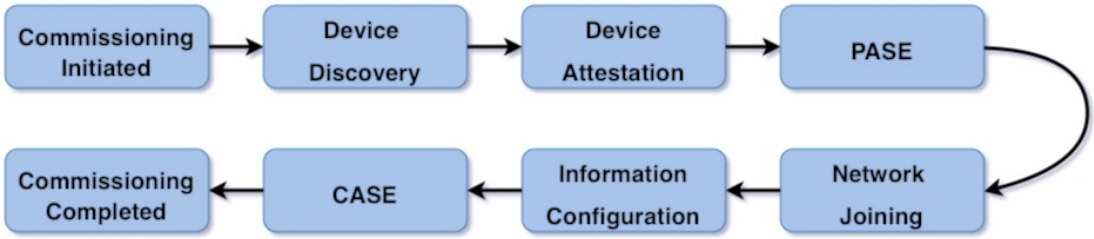


Figure 3.6: Simplified Commissioning Process without Error Handling

Interoperability testing ensures seamless communication between devices from different manufacturers, confirming adherence to Matter specifications and promoting compatibility within the smart home ecosystem. This comprehensive commissioning process ensures that Matter devices are successfully integrated and operational within the smart home, offering efficient and secure connectivity and control.

Once all the above steps are completed successfully, the commissioning can be finalized. This is achieved by exchanging a message, encrypted by a PASE-derived encryption key on the network, indicating the successful commissioning. It is important to note that the process depicted in Figure 3.6 represents a simplified and optimal commissioning process without error handling. In case a critical error occurs during commissioning, the process is aborted, and no further action is conducted.

3.5.1 Device Discovery

As Matter supports a variety of different technologies, device discovery and commissioning can be accomplished using BLE, Wi-Fi, or via IP if the device is already connected to an

IP network. If the device uses Thread, BLE has to be supported as well. Commissioning using BLE uses the Generic Access Profile (GAP) for discovering and connecting to new devices, and the Generic Attribute Profile (GATT) for transmission of credentials, and all three advertising channels should be used.

Before a device's discovery, the announcement of said device has to be completed. A device has to announce in all of its supported technologies, without any restrictions to prioritizing certain technologies. Commissioners on the other hand have to initiate device discovery in all technologies supported by the device that has to be discovered, assuming they are known. Otherwise, it has to initiate discovery in all technologies it supports itself.

The announcement must stop after a maximum duration of 15 minutes, in order to prevent unnecessary pollution of the 2.4 GHz spectrum. However, manufacturers can choose to announce their devices for a shorter period of time if they wish. After the announcement, a device shall not start announcing again for at least 3 minutes.

Upon advertising, the device provides a required 12-bit value for the discriminator, optional fields for Vendor ID and Product ID (each 16 Bit), and a variable sized field for Extended Data (also optional).

3.5.2 Onboarding Payload

The onboarding payload is used in order to allow the onboarding of a new Matter device into an existing Matter network. Various representations can be chosen, allowing for a variety of possible commissioning procedures. The possible representations are: machine-readable formats like QR-Codes and NFC, and human-readable numeric strings interpreted as a manual pairing code.

As a composition of various different parameters, the onboarding payload consists of the following components:

1. **Version:** The version element is a required component, indicating versioning of the payload, and has to be included. Machine-readable formats use 3 bits for versioning, starting with an initial value of `0b000`. For the manual pairing code, just 1 bit is required, starting with `0b0`.
2. **Vendor ID & Product ID:** Represented by two 16-bit values, the Vendor ID and Product ID have to be implemented for machine-readable formats, and can be implemented for human-readable ones. This allows the commissioner to identify the device's model and manufacturer, which can then be used later in the commissioning process.
3. **Custom Flow:** Signals which Device Commissioning Flow shall be used. Consisting of a 2-bit unsigned enumeration, there are three values the parameter can have. A value of `0` indicates the standard Commissioning Flow, where the device will automatically change into advertising mode upon startup. Therefore, no user

interaction is required. A value of `1` signals, that a user-interaction is required before the advertising is going to start. This can, for example, be issued by the press of a button on the device. A value of `2` indicates, that an interaction with a service, provided by the device manufacturer is required for the device setup. Only after this initial setup has been completed, the device is can be commissioned by a commissioner.

4. **Discovery Capabilities Bitmask:** Machine-readable formats require an 8-bit capabilities bitmask. The Bitmask provides important information about the device's available technologies for device discovery.
5. **Discriminator Value:** Consisting of a 12-bit unsigned integer, the Discriminator Value must match the value a device advertises during commissioning. In order to provide the ability to distinguish between different devices, each device should have its own unique discriminator value. Machine-readable formats use the whole 12-bit value, while human-readable formats only use the upper 4 bits of the discriminator.
6. **Passcode:** The passcode element consists of a required 27-bit unsigned integer, which serves as a `Proof of Possession` during the commissioning process. As the 27-bit unsigned integer encodes an 8-bit decimal numeric value, it should be limited to values ranging from `0×0000001` to `0×5F5E0FE`, or `00000001` to `99999998` in decimal. Excluding the invalid passcodes: `00000000`, `11111111`, `22222222`, `33333333`, `44444444`, `55555555`, `66666666`, `77777777`, `88888888`, `99999999`, `12345678`, and `87654321` due to their trivial and insecure nature.
7. **Tag Length Value (TLV) Data:** The TLV is of variable length. It can be included in the payload of machine-readable formats, in order to provide optional information. For further information, see 5.1.5 TLV Content of [19].

Depending on the format, the elements get either packed into a Packed Binary Data Structure (for machine-readable formats), as seen in Figure 3.7, and Base-38 encoded afterward, or, for human-readable formats, just encoded to an 11-digit or 21-digit decimal numeric number string, depending on the presence of the optional Vendor ID and Product ID.

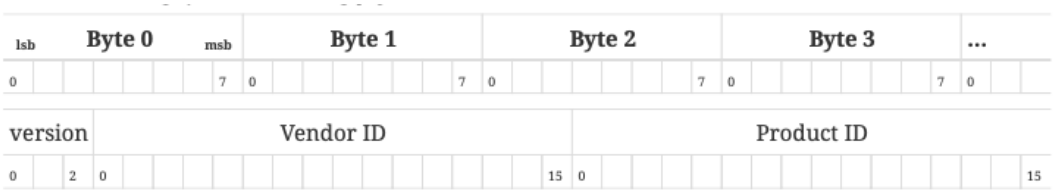


Figure 3.7: Packed Binary Data Structure for Onboarding Payload [19]

3.5.3 Device Attestation

In the aspect of device trustworthiness and integrity, Device Attestation holds significant importance. In the process of Device Attestation, the commissioner cryptographically verifies a commissionee during commissioning. For doing so, a Device Attestation Certificate (DAC), plays a major role. ECDSA with SHA256 is used as the signature algorithm.

Every commissionable Matter device has to include a DAC and corresponding unique private key. To meet the standards, the DAC must adhere to the DER-encoded X.509v3 format as defined in RFC 5280 [1]. It must be issued by a Product Attestation Intermediate (PAI) that is directly linked to an approved Product Attestation Authority (PAA). The certification path length of the DAC should be limited to 2. In order to verify the origin of the certificate, the DAC should include values for Product ID and Verndor ID.

At first, the commissioner generates a random 32-Byte nonce. This nonce gets sent to the commissionee and Attestation Information is requested. The Attestation Information contains a Matter TLV Payload, a signature and a secret Attestation Challenge. Upon receipt, the commissionee has to return the signed Attestation Information to the commsissioner.

For a successful attestation, the commissioner has to validate a bunch of requirements such as:

- The PAA has to be present in the commissioner's trusted root store.
- The sole presence of PAA, PAI and DAC has to be confirmed.
- The Vendor ID of the DAC, has to match the Vendor ID of the PAI certificate.
- In case the PAA certificate contains a Vendor ID, it also has to be identical to the one of the PAI certificate.
- The Device Attestation Signature has to be validated.
- The Attestation Nonce must match the nonce of the commissioner.
- The Certification Declaration signature has to be validated, and the public key has to be obtained from the CSA's Certificate Authority Certificate.
- The Certification Declaration itself has to be validated.

The Certification Declaration (CD) is a cryptographic document that validates the adherence of a Matter device to the protocol standards. Encoded in the Cryptographic Message Syntax (CMS) format, as described in RFC 5652[2], the CD serves as evidence of the device's compliance. Once a device type successfully completes the certification process, the Connectivity Standards Alliance generates a CD specifically tailored for that

[1] https://www.rfc-editor.org/rfc/rfc5280
[2] https://www.rfc-editor.org/rfc/rfc5652

device type. This CD is then provided to the manufacturer for integration into the device firmware.

3.5.4 Information Configuration

In order to allow a Matter node to identify itself within the network, Node Operational Credentials are used, which are set up during commissioning. These Node Operational Credentials consist of a variety of information such as the regulatory domain, UTC time, the NOC and the configuration of network interfaces. Each newly commissioned node receives its NOC from the commissioner using the Node Operational Credentials Cluster (see Section 3.6.1), which is used to manage Node Operational Credentials within a node.

3.5.5 Network Joining

After setting up all required information, the new device gets triggered into connecting to the operational network. This is achieved using the Network Commissioning Cluster, which is used to associate a new node with one or more network interfaces, such as Wi-Fi, Ethernet or Thread. Each instance of the Network Commissioning Cluster can only be applied to a single network interface. Once the node is joined the network, its IPv6 address is discovered for future addressing. This is done using Operational Discovery. Operational Discovery is used, to dynamically discover a node's IPv6 address, without assuming a fixed static IP address for each device.

3.6 Information Model

The information model of Matter provides a detailed and comprehensive representation of the structure, capabilities, and behaviors of Matter-compliant devices within a smart home environment. It establishes a standardized framework that enables interoperability and seamless communication between devices and applications.

At the core of the information model are device types, which categorize different types of smart devices that can be connected within the Matter ecosystem. These device types include lights, switches, Heating, Ventilation and Air Conditioning (HVAC) systems, door locks, sensors, and multimedia devices, among others. Each device type is associated with a predefined set of capabilities that define the actions and functionalities the device can perform. These capabilities are implemented through the concept of Clusters.

Clusters represent functional units that group related commands and attributes together. They provide a standardized and consistent way of defining device capabilities across different manufacturers and models. Clusters encapsulate the features and functions of a specific device type and define the commands that can be issued to control the device as well as the attributes that reflect its current state or properties. For example, a light might have a Lighting Cluster that includes commands for turning the light on or off, adjusting brightness, and changing color.

Endpoints are the communication channels or interfaces through which devices and applications interact within the Matter ecosystem. Each device can have one or more endpoints, and each endpoint can support one or more Clusters. Endpoints represent a logical grouping of Clusters and provide a means to access the capabilities of a device. For example, a smart lightbulb might have an endpoint for lighting control and another endpoint for energy monitoring.

When a Matter controller or application wants to discover the capabilities of a device, it follows a discovery process. It starts by requesting information about the available endpoints of the device. The device responds with a list of endpoints and their associated Clusters. The controller then sends requests to each endpoint individually to further discover the specific Clusters and their supported commands and attributes.

The information model ensures that different devices, regardless of their manufacturer or model, can communicate and be controlled in a consistent and interoperable manner. It enables seamless integration of devices into a unified smart home ecosystem, where applications and controllers can easily discover and interact with devices, leveraging their capabilities to create personalized and intelligent automation experiences.

3.6.1 Clusters

Clusters play a central role in facilitating interoperability and defining the functionality of devices within the ecosystem. They can be seen as logical groups of related capabilities or features that devices can support. Matter Clusters provide a standardized way of organizing and categorizing device functionality, enabling seamless communication and interaction between devices from different manufacturers. Each cluster is associated with specific device capabilities or behaviors, such as lighting control, temperature sensing, or door lock functionality. Clusters define a set of attributes, commands, and events that devices within the cluster can utilize to exchange information and perform actions. This standardized structure ensures that devices supporting the same cluster can effectively communicate and interoperate with one another, regardless of the manufacturer, and are addressed by their unique Cluster IDs.

Matter defines two types clusters are divided into, `Server Clusters` and `Client Clusters`. A `Server Clusters` is stateful, meaning it holds attributes, events, and commands. A `Client Cluster`, on the other hand, is stateless, meaning it does not save any information but rather initiates interactions of a `Server Cluster`. Each `Client Cluster` can be bound to one or more `Server Clusters`, therefore enabling control over multiple devices simultaneously. A simplified overview of a possible setup consisting of two lamps can be seen in Figure 3.8. In this example, both lamps implement the `ON/OFF Light` device type, meaning they have to include the `ON/OFF Server Cluster` which is responsible for controlling the light. However, similar to classical table lamps, the devices also include an `ON/OFF Light Switch` device type, responsible for the physical switching of the light. This switch has to implement the `ON/OFF Client Cluster`, which may control other `ON/OFF Server Cluster`. The switch of Lamp

A is bound to its own lamp, and the one of Lamp B, and therefore able to control both of them, while the switch of Lamp B can only control its own associated lamp.

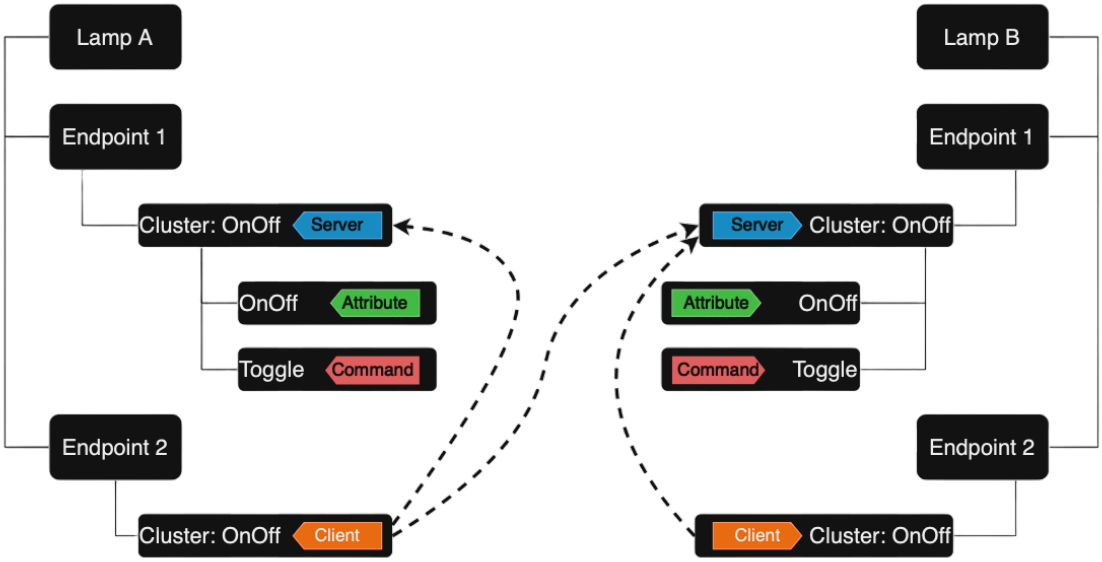


Figure 3.8: Information Model Overview, adapted from [21]

For further clarification of the information model's structure, this section will describe the requirements set by a simple Matter Light bulb (with the device type `ON/OFF Light`). Upon the discovery, the controller retrieves a device's Descriptor Cluster, responsible for maintaining lists of relevant information over a device, such as device types, implemented Clusters, and associated endpoints. In the case of an `ON/OFF Light`, which is a lighting device that can be switched `ON` or `OFF` by a connected controller like a switch or dimmer, the specification of the `ON/OFF Light` device type shows that each device labeled as such must implement a set of mandatory clusters. These clusters include:

- **Identify** — Provides functionality to identify a physical device to an observer, such as a blinking LED, audible beeping, and many more depending on the device's hardware.
- **Groups** — Manages a node-wide group table per endpoint, allowing devices to be grouped together and controlled with a single command instead of addressing each node separately.
- **Scenes** — Provides attributes and commands for setting up and recalling scenes. Scenes are predefined structures with a set of stored values, designed to handle specific workflows. For example, a scene could turn on a bathroom light and dehumidifier once someone enters a bathroom at a certain time of the day.

- **On/Off** — Handles turning a device on and off by setting an internal variable.
- **Level Control** — Offers a means to control a specific aspect of a device, which can be adjusted to a certain level. For instance, it allows you to manage the brightness of a light, the extent of a door's closure, or the power output of a heater.
- **Occupancy Sensing** — Provides a user interface for occupancy sensing features, which includes the ability to configure settings and receive notifications regarding the occupancy status.

Even though the device does not have to practically implement clusters like the Level Control cluster, it must accept corresponding commands. Therefore, even if the light cannot be dimmed, it has to handle commands trying to change its level, having no effect, to provide a constant user experience.

Clusters are designed to be extensible and flexible, allowing for the inclusion of additional attributes and commands to accommodate future advancements in smart home technology. This scalability ensures that the Matter standard can adapt to evolving needs and support new device functionalities as they emerge.

Once the Clusters associated with each endpoint are known, the controller can retrieve detailed information about the capabilities and attributes of the device. This includes querying the supported commands, reading the attributes' current values, and subscribing to event notifications.

3.6.2 Endpoints

An endpoint refers to a logical entity within a device that provides specific functionalities or features. It represents a distinct capability or service that can be accessed and controlled by a Matter controller or other devices within the fabric.

Each endpoint is associated with one or more Clusters, which define the commands, attributes, and events related to the specific functionality provided by the endpoint. Clusters are organized within endpoints to categorize and group related capabilities together. Endpoints allow for granular control and management of device functionalities. For example, a smart light bulb may have multiple endpoints, each representing a different aspect of control, such as brightness, color, or power. Each of these endpoints will have its own set of Clusters, handling further control over the device. A simplified overview can be seen in Figure 3.8.

When a Matter controller discovers a device, it retrieves information about the available endpoints associated with that device. In the case of an `ON/OFF Light` device, the controller will discover endpoints for Identify, Groups, Scenes, On/Off, Level Control, and Occupancy Sensing. By understanding these endpoints and their associated Clusters, the Matter controller can interact with the `ON/OFF Light` device at a more detailed level, issuing specific commands, reading attribute values, and subscribing to event notifications.

CHAPTER 4

Web of Things

The WoT expands the IoT by integrating Web technologies, enabling seamless collaboration between physical devices, digital services, and the World Wide Web (WWW). Embedded sensors, actuators, and communication capabilities equip everyday objects with the ability to collect and share data, perform intelligent actions, and interact with users through Web-based interfaces. Standardized protocols and open Web standards ensure interoperability, accessibility, and scalability across diverse devices and platforms, fostering a unified ecosystem. Despite increased connectivity, the IoT landscape remains fragmented at the Application Layer, making application development challenging and integration of smart things into composite applications difficult. Various service platforms have emerged to address this issue, but they often lack compatibility and are limited to a small community of expert developers. In contrast, the Internet demonstrates a scalable network of interoperable computers, and the WWW showcases the power of simple and open standards for creating flexible systems. The WWW's integration capabilities and ubiquitous availability on different devices make it an ideal universal integration platform. WWW resources offer not only Web pages but also Application Programming Interfaces (APIs) that enable the development of composite applications accessible from desktops and mobile devices. [22, 23]

4.1 Thing Descriptions (TDs)

The introduction of TDs has been instrumental in establishing a standardized and structured approach to describing and accessing IoT devices and services. By serving as a bridge between applications and IoT devices, TDs facilitate seamless integration and interoperability across a wide range of platforms and environments. Their significance lies in providing a unified format for describing the capabilities, interfaces, and metadata of IoT devices within the WoT framework, thereby fostering a cohesive ecosystem.

TDs play a crucial role in enabling interoperability by ensuring that devices from different manufacturers, with varying functionalities and communication protocols, can work together harmoniously. By adhering to the standardized format, TDs provide a common language for devices and applications to communicate effectively, irrespective of their individual characteristics. This standardized approach streamlines the integration process, reducing complexity and effort when incorporating new devices into existing systems or developing applications that interact with multiple IoT devices. TDs also enable extensibility and scalability in the IoT domain. The structured format allows for the inclusion of additional information, such as semantic annotations, to provide contextual meaning and enhance the richness of device descriptions. Semantic annotations enable devices and applications to leverage shared vocabularies, ontologies, and domain-specific knowledge, resulting in more intelligent and context-aware interactions. This extensibility empowers developers to accommodate evolving requirements and leverage emerging technologies within the WoT.

4.2 Structure of Thing Descriptions

The purpose of this chapter is to provide a condensed but detailed look at the underlying information model of the WoT, namely TDs, which was derived from the official Web of Things (WoT) Thing Description 1.1 [24] documentation.

By investigating TDs, developers and applications can easily discover and understand the capabilities and interfaces of IoT devices, regardless of their underlying technology or manufacturer. This standardization promotes interoperability, simplifies integration, and enables seamless communication and interaction between diverse devices and services within the WoT ecosystem.

A TD is structured using the JavaScript Object Notation for Linked Data (JSON-LD) format, which allows for the representation of linked data using JSON syntax. In order to visualize the whole structure, an example TD, adapted from [24], for a simple Light Bulb can be seen here:

```
{
  "@context": "https://www.w3.org/2022/wot/td/v1.1",
  "title": "Light Bulb",
  "description": "A simple Light Bulb, capable of \
  toggling and reporting its status. \
  In case of overheating an event is triggered.",
  "securityDefinitions": {
      "basic_sc": {"scheme": "basic", "in": "header"}
    },
  "security": "basic_sc",
  "properties": {
      "status": {
          "type": "string",
          "forms": [{"href": "https://lamp.example.com/status"}]
```

```
        }
    },
    "actions": {
        "toggle": {
            "forms": [{"href": "https://lamp.example.com/toggle"}]
        }
    },
    "events": {
        "overheating": {
            "data": {"type": "string"},
            "forms": [{
                "href": "https://lamp.example.com/oh",
                "subprotocol": "longpoll"
            }]
        }
    }
}
```

The TD provides a machine-readable description of a `Thing`, refering to an IoT device or entity that is part of the interconnected network and representing a physical or virtual object that can be accessed, controlled or monitored, in the WoT ecosystem, specifying its capabilities, properties and actions and is structured as follows:

- **Context**: represents an array that defines the vocabulary used in the TD. It provides a standardized way to specify the meaning of terms and properties used within the TD. The Context field is an essential part of the JSON-LD format and plays a crucial role in enabling semantic interoperability between different systems and applications. Using the Context field, applications consuming the TD can resolve the meaning of terms and properties without ambiguity. They can understand the types, units, and semantics associated with properties and actions, enabling them to interact with `Things` in a meaningful way.
- **Title**: provides a human-readable name or title for the `Thing`. It allows developers and users to quickly identify and refer to the `Thing` using a descriptive name.
- **Description**: provides a textual description of the `Thing`. It allows developers, users, and other stakeholders to understand the purpose, functionality, and characteristics of the `Thing` in detail.
- **Security**: is used to specify the security mechanisms and requirements associated with accessing and interacting with the `Thing`. It provides important information about the security considerations that need to be taken into account when communicating with the `Thing`. The Security field includes one or more security schemes or mechanisms (which are defined in the `Security Definitions` field), that can be either conventional ones, like the use of a Bearer Token or pre-shared key,

or custom-defined. Each security scheme provides details about how to secure the communication and access to the `Thing`'s resources.

- **Security Definitions**: used to define the security schemes or mechanisms that can be employed for accessing and interacting with the `Thing`'s resources. It provides a way to describe the details of each security scheme, including its properties and configurations. The field holds an object where each property represents a security scheme name, and its value describes the details of that security scheme. The structure and properties of each security scheme can vary depending on the specific scheme being used.
- **Properties**: used to describe the properties of the `Thing`. Properties represent the state or characteristics of a `Thing` that can be observed, queried, or modified. The field is represented as an object within the TD, where each property is defined as a key-value pair. The key represents the name or identifier of the property, while the value provides additional details and constraints related to that property.
- **Actions**: used to define the actions or operations that can be performed on the `Thing`. Actions represent the behaviors or functionalities that can be invoked or executed on the `Thing`. The field is represented as an object within the TD, where each action is defined as a key-value pair. The key represents the name or identifier of the action, while the value provides additional details and constraints related to that action.
- **Events**: used to define the events or notifications that can be emitted or received by the `Thing`. Events represent occurrences or changes that can be observed or subscribed to by interested parties. Again the field is represented as an object within the TD, where each event is defined as a key-value pair. The key represents the name or identifier of the event, while the value provides additional details and constraints related to that event.
- **Forms**: describes the different interaction patterns or protocols. This field in a TD consists of an array that contains different forms or ways to interact with the `Thing`. Each form represents a specific interaction pattern or protocol that can be used to access or manipulate the `Thing`'s resources.

CHAPTER 5

Implementation

As Matter presents a significant opportunity to advance the capabilities of interconnected systems, this chapter aims to explore the seamless integration of Matter devices into a Matter network, through a proof of concept. The primary objective is to set up a Thread Border Router, connected with its associated Radio Co-Processor (RCP) leading the network, and connect two nRF52840 Development Kits[1], acting as a Matter Light Bulb and a Matter Light Switch. By implementing this proof of concept, the focus lies on constructing a network infrastructure utilizing the Matter protocol and conducting a simple binding between the connected devices. A simplified overview of the process can be seen in Figure 5.1. Once everything is set up, and the devices bound together, each device is mapped into a TD (see Section 5.2). Before this mapping can be achieved, a PB has to be developed, which defines crucial parameters of the mapping.

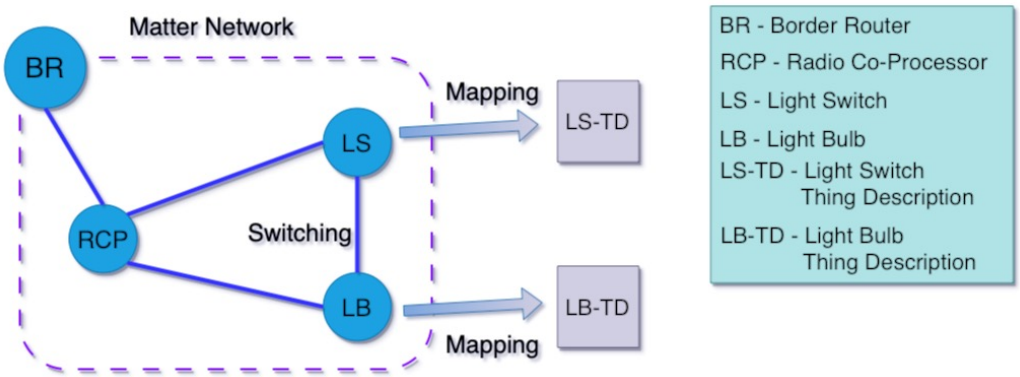


Figure 5.1: Proof of Concept Overview

[1] https://www.nordicsemi.com/Products/Development-hardware/nRF52840-DK

5.1 Proof of Concept

This proof of concept explores the implementation of a Matter network, running on Thread, and the integration of Matter devices. The focus is on constructing a network infrastructure using the Matter protocol and establishing a simple binding mechanism between the devices to enable seamless communication and interoperability. The aim is to showcase the potential of the Matter network and its ability to create a cohesive and efficient ecosystem of interconnected devices.

5.1.1 Border Router

Before any device can be connected to the Thread network, a Border Router, precisely an OpenThread Border Router (OTBR) has to be set up. The OTBR creates the whole Thread network and handles communication between said network and an external Wi-Fi network. Therefore, the OTBR acts as a bridge, facilitating communication and protocol translation between the Thread network and the Wi-Fi network. In order to set up the Border Router a Raspberry Pi 3B[2] or newer is required. The Raspberry Pi is flashed with the Raspberry Pi OS and booted up. Once the device is booted up and connected to the Internet via Ethernet, the OTBR repository has to be cloned and initialized using the following commands in the terminal:

```
git clone https://github.com/openthread/ot-br-posix
cd ot-br-posix
./script/bootstrap
```

Once the initialization is completed, a network interface has to be chosen and parsed to the setup script:

```
  INFRA_IF_NAME=eth0 ./script/setup
```

If the setup was successful, a RCP has to be connected to the Border Router via USB. This RCP is resonsible for communication and management within the future Thread network, and can be created by flashing a nRF52840 DK with the corresponding RCP-firmware. This can be done by completing the following steps:

```
  cd ~/src
  git clone --recursive \
  https://github.com/openthread/ot-nrf528xx.git
  cd ot-nrf528xx
  script/build nrf52840 USB_trans
  cd ~/src/ot-nrf528xx/build/bin
  arm-none-eabi-objcopy -O ihex ot-rcp ot-rcp.hex
```

[2] https://www.raspberrypi.com/products/raspberry-pi-3-model-b/

After connecting the development board to the flasher its serial port name can be found using:

```
ls /dev/ttyACM*
```

This may result in a serial port name like /dev/ttyACM0, depending on the used USB slot and other connected devices. For flashing the board, the nRF5x Command Line Tools[1], SEGGER J-Link [2] drivers, and its serial number are required, which can be found on the board itself. Finally, the development board can be flashed using these commands in the terminal:

```
cd ~/nrfjprog/
./nrfjprog -f nrf52 -s <SERIAL_NUMBER>  --verify --chiperase\
--program ~/src/ot-nrf528xx/build/bin/ot-rcp.hex --reset
```

The, now successfully flashed, RCP can now be connected to the Border Router using USB. After finding its serial port name, like described earlier, the OTBR's configuration, found at /etc/default can be changed accordingly by replacing the default serial port name with the actual one. After updating the value, the device can be restarted and should boot up with all required services available. This can be verified using systemctl and looking for mdns.service, otbr-agent.service, and otbr-web.service.

The OTBR's Web interface can now be reached via its assigned IP address, in order to form a new Thread network, and the development boards can be connected to the fabric.

5.1.2 Development Boards

Before the development boards can be used as Matter devices and therefore connected to the fabric, they have to be flashed with the corresponding firmware as well. This proof of concept is limited to a two-device setup, namely a Matter Light Bulb[1] and a Matter Light Switch[2]. Both devices have available example—implementations, provided by Nordic Semiconductors[3]. Using the nRF Connect for VS Code plugin for Visual Studio Code, the example applications, namely the light bulb and switch, can be created and built. Once the building process is finished, the boards can be flashed through the provided Graphical User Interface (GUI). After flashing the devices, device advertising can be enabled by pressing the physical button Button 4 on the devices. Commissioning

[1] https://www.nordicsemi.com/Products/Development-tools/nrf-command-line-tools/download

[2] https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack

[1] https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/samples/matter/light_bulb/README.html

[2] https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/samples/matter/light_switch/README.html

[3] https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/samples/matter.html

the devices into the Matter fabric requires a Matter controller capable of discovering the boards over BLE.

5.1.3 Communication Testing

The Matter controller, in theory, can be realized by any device capable of supporting BLE and Wi-Fi that runs on a 64-Bit architecture, even though only one system would be able to work properly. For this proof of concept, macOS 12.6.5, running on an iMac (late 2015), was used to function as the Matter contoller. Setting up the Matter controller is rather straightforward, at first, the corresponding repository has to be cloned using the following command:

```
git clone https://github.com/project-chip/connectedhomeip
```

Once the repository is cloned successfully, the CHIP-Tool has to be built from it. This can be achieved using:

```
cd connectedhomeip
./scripts/examples/gn_build_example.sh \
examples/chip-tool BUILD_PATH
```

where BUILD_PATH resembles the destination the binaries should be built to. Now that everything is set up, commissioning the devices can start. In order to commission a new device into the Thread network, its network credentials are needed. Those credentials can be obtained by connecting to the Border Router and running this command:

```
sudo ot-ctl dataset active -x
```

For this proof of concept the credentials, in hex format, are:

```
0e080000000000010000000300000f35060004001\
fffe00208111111122222220708fd664a1b0937\
791705100011223344556678899aabbccddeeff0\
30b4d617474657220546573740102123404 10b59e\
e29b5f4c21de4fe69eebd65bbe500c0402a0f7f8
```

After finding the network credentials commissioning can start by pairing the first device using this command:

```
./chip-tool pairing ble-thread 1 \
hex:NETWORK_CREDENTIALS 20202021 3840
```

The string NETWORK_CREDENTIALS has to be replaced with the credentials obtained in the last step. 20202021 and 3840 are preset values that represent the device's

PIN-Code and discriminator, which are set to default values for these examples. 1 represents the node ID that will be assigned to the newly commissioned device upon success.

After commissioning the device, the controller prints the message "Secure Session to Device Established" which signals the success of the process. Using the same command for the second device, just changing the 1 to 2, the commissioning can be completed. The Light Bulb has been configured to be represented by node ID 1, while the Switch was assigned node ID 2.

In order to enable controlling the light bulb, using the switch, proper Access Control List (ACL) has to be added to the bulb. This can be accomplished using the following command:

```
./chip-tool accesscontrol write acl '[{"fabricIndex": 1, \
"privilege": 5, "authMode": 2, "subjects": [112233], \
"targets": null}, {"fabricIndex": 1, "privilege": 3, \
"authMode": 2, "subjects": [2], "targets": [{"cluster": 6, \
"endpoint": 1, "deviceType": null}, {"cluster": 8, \
"endpoint": 1, "deviceType": null}]}]' 1 0
```

This command provides the Switch with the necessary permissions to control the Light Bulb. The last step consists of creating a binding table to the Switch, in order to notify the Light Bulb once the Switch gets activated. This can be achieved using:

```
./chip-tool binding write binding '[{"fabricIndex": 1, \
"node": 1, "endpoint": 1, "cluster": 6}, {"fabricIndex": 1, \
"node": 1, "endpoint": 1, "cluster": 8}]' 2 1
```

Now that all devices are bound and ready for communication, Button 2 can be pressed, on the Switch, and as a result, the internal LED of the Light Bulb gets toggled between ON and OFF. Therefore, the setup was successful and communication between the two development boards could be established.

5.2 Thing Descriptions

Establishing a connection to the Matter network (see Section 5.1) is crucial before deriving valid TDs from the selected Matter devices, such as the light bulb and the light switch. This initial connection ensures that the devices are recognized and accessible within the network, allowing for effective communication and data exchange.

Once this requirement is fulfilled mapping of the devices can begin. As the TDs of the light bulb and the light switch are mostly identical, this section will focus primarily on the light bulb as it is the more complex device. Furthermore, for the sake of simplicity,

not all of the device's attributes will be mapped, but only the ones relevant for controlling the devices from the WoT.

As explained in Section 4 each TD consists of different static and dynamic components. Fields like the `Security` or `ID`, are static as they do not change during normal operation. Therefore, these fields can already be mapped into the TD. The device's IPv6 address serves as the ID, and as Matter establishes sessions using a password (see Section 3.3), `psk_sc` can be used as the security scheme. The `Title` and `Description` fields are optional, but provide human readers with handy information. All this results in the following provisional TD:

```
{
        "@context": "https://www.w3.org/2019/wot/td/v1",
        "title": "Matter Light Bulb",
        "id": "ff35:0040:2941:3b00:f5e2:c80e:94a2:6a8c",
        "description": "Light Bulb Example",
        "securityDefinitions": {
                "psk_sc": {
                        "scheme": "psk"
                }
        },
        "security": "psk_sc",
        "properties": {},
        "actions": {},
        "events": {}
}
```

To map the remaining fields into the TD within the WoT, a PB specific to the Matter protocol must be developed. This PB acts as a mediator, enabling seamless integration and accurate representation of Matter devices within the WoT ecosystem. It bridges the communication gap and ensures interoperability between Matter devices and WoT applications or services.

5.2.1 Protocol Binding

As the WoT does not provide a PB for Matter, one has to be developed in order to enable control over Matter devices via the WoT using TDs. A PB refers to the mapping or adaptation of the WoT interfaces and interactions to a specific communication protocol. It defines how WoT devices and services communicate and exchange data over the network. The PB, developed in this section, defines how the concepts of the WoT, such as Properties and Actions, are mapped to Matter.

As the WoT is based on Web technology, the PB has to provide a suitable Uniform Resource Locator (URL) format for communication with Matter. The URL format developed in this thesis looks like this:

```
matter://{deviceAddress}/{clusterName}\
/{functionName}/{nodeID}/{endpointID}
```

For a more comprehensive understanding and detailed information regarding the parameters mentioned above, refer to Table 5.1, where further clarification and elaboration are provided.

Term	Description	Mandarory	Type
deviceAddress	Represents the devices IPv6 address	yes	IPv6
clusterName	Defines the cluster that shall be addressed	yes	String
functionName	Is the name of the function that should be called	yes	String
nodeID	Represents the device's Node ID, assigned during commissioning	yes	Integer
endpointID	Defines the endpoint the functionality is instantiated on, usually 1	yes	Integer

Table 5.1: URL Parameters of the Protocol Binding

These parameters play a crucial role in providing the Matter device with most of the necessary information required to successfully execute the intended operation. It is important to note, however, that before the PB can be effectively utilized, additional form terms must be provided to ensure the seamless integration of the Matter device into the WoT. Form terms provide a structured way of defining inputs and interactions for a Thing within the WoT ecosystem and are defined in Table 5.2.

Term	Description	Mandarory	Type
matter:level	Describes the desired brightness level of a dimmable light	no	Integer
matter:transitionTime	Defines the time the transition from the old brightness level to the new one should take in milliseconds	no	Integer
matter:attributeValue	The value of the attribute defined in the URL should be set to	no	Integer
matter:atributeName	Defines the desired attribute the functions should access	no	String

Table 5.2: Form Terms of the developed Protocol Binding

With the crucial definitions of all the necessary parameters now in place, the next significant step in the process unfolds, namely initiating the derivation of the Matter device's attributes.

5.2.2 Attributes

By utilizing the PB defined in Section 5.2.1, deriving the relevant attributes is rather straightforward. Once again the CHIP Tool was used to obtain a list of available attributes during operation of the Matter light bulb. This could be achieved by using the following

command, which reads from the `Descriptor` cluster of the node with Node ID `1` and endpoint `1`:

```
./chip-tool descriptor read attribute-list 1 1
```

Execution of the above command results in a long output containing the list of available attributes, displayed as their associated attribute ID in decimal format looking like this:

```
Endpoint: 1 Cluster: 0x0000_0006 Attribute 0x0000_FFFB \
DataVersion: 3255845681
AttributeList: 5 entries
[1]: 0
[2]: 16384
[3]: 16385
[4]: 16386
[5]: 16387
```

By skimming through the available endpoints of the device a comprehensive list of all available attributes can be obtained. However, as these attributes are only referenced by their IDs, they are not of much use. In order to figure out which attributes lie behind those numbers, the Matter Cluster Specification [25] has to be consulted. After searching the document for the specific cluster ID `0×0000_0006`, the `On/Off` cluster is discovered. The specification holds a list of attributes the cluster manages sorted by their attribute IDs. By comparing those IDs with the ones obtained through the command earlier, the attribute names can be identified. The ID `0` for example, corresponds with the `OnOff` attribute of the cluster, responsible for managing the devices state. This way all available attributes can be obtained and derived into TD properties.

The following example will show the mapping of the `OnOff` attribute into the TD, the identical process can be applied to the remaining attributes afterwards. As the communication is only unilateral, the `observable` field, responsible for subscribing to changes of the fields by WoT clients, has to be set to `false`.

```
"properties": {
    "onoff": {
      "title": "onoff",
      "observable": false,
      "readOnly": true,
      "type": "boolean",
      "forms": [
        {
          "op": [
            "readproperty"
          ],
          "href": "matter://ff35:0040:2941:3b00:f5e2:\
```

```
            c80e:94a2:6a8c/onoff/read/1/1"
          }
        ]
      }
    }
```

After completing the attribute mapping process, the focus shifts towards deriving the relevant functions of the Matter device.

5.2.3 Actions

Following roughly the same procedure, a Matter device's functions can be discovered by using the CHIP Tool. The only difference lies in the underlying structure, each cluster is categorized as either a server cluster or a client cluster. While server clusters are stateful and manage attributes, events and functions, a client cluster is stateless, which means it only initiates functions. Extracting those clusters, and subsequently, the available functions, can be achieved by using the following command, which reads from the `Descriptor` cluster of the node with Node ID `1` and endpoint `1`:

```
./chip-tool descriptor read client-list 1 1
./chip-tool descriptor read server-list 1 1
```

Execution of the above command results in a long output containing the list of available clusters, displayed as their associated cluster IDs in decimal format looking like this:

```
Endpoint: 1 Cluster: 0x0000_001D Attribute 0x0000_0001 \
DataVersion: 738870138
ServerList: 4 entries
[1]: 3
[2]: 4
[3]: 6
[4]: 8
```

By skimming through the available endpoints of the device a comprehensive list of all available clusters can be obtained. However, as these clusters are only referenced by their IDs, they are not of much use at the moment. In order to figure out which clusters lie behind those numbers, the Matter Cluster Specification [25] has to be consulted. After searching the document for the specific cluster IDs, the cluster names are revealed. The ID `3` corresponds with the `Identify` cluster, `4` with the `Groups` cluster, `6` with the `On/Off` cluster and `8` with the `Level Control` cluster.

The specification holds a list of functions that those clusters implement. The `On/Off` cluster, for example, implements the relevant functions: `off`, `on` and `toggle`. Mapping those functions into a WoT compatible format is rather straightforward. Contrary

to the process of mapping attributes, functions may need additional parameters to function. Such a function is implemented by the `Level Control` cluster and is called `move-to-level`. It provides the Matter device with a desired brightness level and the time within which the transition should take place. This function could be mapped like this:

```
"actions": {
    "move-to-level": {
      "title": "move-to-level",
      "forms": [
        {
          "op": [
            "invokeaction"
          ],
          "href": "matter://ff35:0040:2941:3b00:f5e2:\
          c80e:94a2:6a8c/levelcontrol/move-to-level/1/1",
          "matter:level": 100,
          "matter:transitionTime": 500
        }
      ]
    }
  }
```

In this example, the Matter device would, upon receipt of the action, transition its brightness level from its current state to `100` within `500 milliseconds`. Following this example, all other relevant functions of the light bulb can be mapped into the TD. The complete mapping and the resulted TDs of both, the light bulb and the light switch can be seen in Section B of the appendix.

5.3 Evaluation

This section presents a comprehensive overview and evaluation of the results obtained in the thesis, which assesses the feasibility and functionality of the Matter standard in addressing the prevalent interoperability and standardization challenges within the smart home industry. The thesis aimed to achieve several key objectives, including the successful implementation of a Matter-enabled smart home system, the development of a PB to facilitate the seamless integration of Matter devices into the WoT ecosystem, and the provision of valuable insights into the setup and utilization of a Matter network.

To commence the thesis, a systematic literature review was conducted, involving an extensive search for academic papers, technical reports, and industry publications. This diligent review process ensured that a concise understanding of existing smart home technologies related to Matter was acquired, thereby guaranteeing that all relevant information was gathered from reliable and authoritative sources.

Drawing upon the official specifications of Matter and the WoT, the thesis proceeded to develop a simplified yet comprehensive overview of the Matter standard, incorporating all the necessary formal structures required for the successful development of the PB. This formal methodology ensured that the implementation of the PB and the mapping of Matter devices into TDs adhered accurately to the guidelines and standards specified by Matter and the WoT.

To validate the feasibility and functionality of the Matter standard, the thesis followed the latest Matter documentation and conducted a comprehensive proof of concept. The primary objective of this proof of concept was to establish a Matter network using nRF52840 Development Kits and a Raspberry Pi, simulating the behavior of a Matter light bulb and light switch. The interaction between these emulated devices was meticulously examined, with a strong emphasis on verifying the seamless control of the light bulb using the switch within the Matter ecosystem. This involved careful observation of the communication between the emulated devices, ensuring that all intended commands and actions were executed correctly. The successful completion of this proof of concept, which confirmed the seamless control of the Matter light bulb through the emulated light switch, demonstrated the practical feasibility of the Matter standard and paved the way for enhanced communication between Matter and the WoT.

Simultaneously, the PB was developed by meticulously considering Matter's core functionalities and formats. This formal approach ensured that the PB aligned harmoniously with the standards set by the Matter standard. Furthermore, the thesis effectively applied these formats and conventions to the WoT information model, resulting in the successful construction of TDs for both the Matter light bulb and the light switch.

Once the TDs were derived successfully, a meticulous validation process was undertaken using the WoT JSON Schema. The primary purpose of this thesis was to systematically examine and validate the integrity, accuracy, and compliance of the defined TDs within the WoT ecosystem. Leveraging the JSON Schema [1] as a robust validation mechanism, the thesis aimed to ensure that the TDs adhered precisely to the specified structure, data types, and constraints, thereby guaranteeing their overall reliability and effectiveness. The utilization of an online `JSON Schema Validator` [2] allowed for a meticulous validation process, with careful attention given to the separate components of the TDs. The TDs successfully passed this validation process, serving as a strong indication that they conform flawlessly to the defined schema.

Moreover, the TDs were subjected to further examination to verify their compliance with the WoT specifications. Each TD included the required `@context` field, which appropriately referenced the relevant context URL for the WoT TD version 1.0. The metadata fields, such as `title`, `id`, and `description`, were meticulously defined in both TDs, providing essential information about the respective devices. The TDs also

[1] `http://json-schema.org/draft-07/schema#`
[2] `https://www.jsonschemavalidator.net`

defined a security scheme named psk_sc utilizing the Pre-Shared Key authentication scheme, thus demonstrating the devices' support for a robust security mechanism.

The properties defined in the TDs exhibited appropriate attributes, including title, type, and forms. Each property precisely specified its accessibility, whether observable or readonly, and associated operations, such as readproperty and writeproperty. These properties aligned seamlessly with the expected characteristics of Matter devices, covering essential aspects such as identification, On/Off functionality, and dimming.

Moreover, the TDs included a set of actions, each of which described a specific behavior or functionality supported by the device. These actions were appropriately titled and accompanied by informative descriptions, enabling users to grasp their intended purpose effectively. The forms section for each action defined the supported operation (invokeaction) and specified the corresponding endpoint for interaction, ensuring a clear and standardized approach.

In summary, the thesis successfully achieved its defined goals by providing a concise understanding of smart home technologies related to Matter through an in-depth systematic literature review. A proof of concept was conducted, showcasing the seamless control of a Matter light bulb using a switch within the Matter ecosystem. Additionally, a comprehensive PB was developed, guaranteeing the accurate implementation of Matter devices into TDs within the WoT ecosystem. The TDs were meticulously validated using the WoT JSON Schema, thereby confirming their integrity and compliance. Ultimately, the thesis effectively demonstrated the feasibility and functionality of the Matter standard, addressing crucial interoperability and standardization challenges within the smart home industry and providing a solid foundation for enhanced control over Matter devices by WoT-enabled devices.

CHAPTER 6

Conclusion and Future Work

In conclusion, this thesis has extensively explored the potential of the Matter standard in addressing the critical challenges related to interoperability among various smart home technologies. The research conducted has not only provided valuable insights into the capabilities of the Matter standard but has also presented a comprehensive proof of concept to demonstrate its practical application.

The primary objective of this thesis was to investigate the potential of the Matter standard in overcoming the barriers to interoperability and integration in smart home ecosystems. To achieve this goal, a comprehensive proof of concept was developed, focusing on a basic smart home scenario involving a light bulb and a light switch. The aim was to showcase how the Matter standard could facilitate seamless communication and control between these devices. The proof of concept successfully demonstrated the capabilities of the Matter standard in establishing a cohesive and interoperable smart home network. By configuring the light bulb and light switch to comply with the Matter standard, they were able to seamlessly interact with each other. The switch effortlessly controlled the functionality of the bulb, highlighting the potential of the Matter standard to bridge the gap between disparate smart home devices and enable effortless integration.

Furthermore, this thesis contributed to the development of a mapping mechanism for Matter devices into the WoT, an emerging paradigm that aims to facilitate interoperability and interaction between IoT devices. By creating a standardized representation of Matter device capabilities and functionalities within the WoT framework, the thesis laid the foundation for enabling WoT devices to control and interact with Matter devices. This mapping mechanism not only enhances interoperability but also paves the way for the seamless integration of Matter devices into larger smart home networks that leverage the power of the WoT.

While the proof of concept and the mapping mechanism represent significant milestones in the exploration of the Matter standard's potential, it is crucial to acknowledge the

limitations and areas for future research. The practical implementation of the developed proof of concept and its real-world application was beyond the scope of this thesis. Future work should focus on translating the theoretical approach into a practical proof of concept, ensuring the direct controllability of Matter devices through the WoT. Such an implementation would provide valuable insights into the feasibility and usability of the proof of concept in real-world scenarios, further validating its effectiveness and potential impact.

Additionally, future research endeavors should center around the development of an algorithmic approach for automatically deriving TDs from Matter devices. The current manual process of mapping relevant information from the Matter Specifications into TDs can be time-consuming and prone to errors. An algorithmic solution would streamline and expedite this process, enabling the efficient generation of TDs for Matter devices. This advancement would be pivotal in driving the adoption and widespread use of the Matter standard, as it would simplify the integration and interoperability of a wide range of smart home devices within the Matter ecosystem.

This thesis represents a substantial scientific contribution to the field of smart home technologies by extensively exploring the potential of the Matter standard, demonstrating its capabilities through a compelling proof of concept, and developing a mapping mechanism for Matter devices into the WoT, serving as foundations for future advancements and practical implementations.

APPENDIX A

Message Format

Name	Length	Required	Description
Message Length	2 Bytes	No	Describes the length of the whole message (not including this field) in Bytes. Must only be specified if the message gets transferred over a stream-oriented channel.
Message Flags	1 Byte	Yes	Consists of 3 subfields, positioned in chronological order: DSIZ Field (2 Bits, Position 0–1) specifies the size and purpose of the `Destination Node ID` (0 = not present, 1 = 62-Bit Node ID present, 2 = 16-Bit Group ID present, 3 = reserved for the future) S Flag (1 Bit, Position 2) indicates if the `Source Node ID` is present Version (4 Bits, Position 4–7) specifies the version of the Matter Message Format
Session ID	2 Bytes	Yes	Identifies the with the message associated session. This is used to find the exact key used to encrypt the message
Security Flags	1 Byte	Yes	Consists of 4 subfields, positioned in chronological order (3 Bits are reserved for future use): Session Type (2 Bits, Position 0–1) specifies the session Type (0 = Unicast Session, 1 = Group Session, 2–3 = reserved for future use). MX Flag (1 Bit, Position 5) indicates that the `Message Extension` is present. C Flag (1 Bit, Position 6) indicates that the message is a control message. P Flag (1 Bit, Position 7) indicates that the message is encoded with privacy enhancements.
Message Counter	4 Bytes	Yes	Unsigned integer value identifying each message. The counter gets incremented for each new message created.
Source Node ID	0/8 Bytes	No	Identifies the Node ID that sent the message.
Destination Node ID	0/2/8 Bytes	No	Describes the unique identifier of the receiving node.
Message Extensions	variable	No	Provides backwards compatible extensibility. The first 2 Bytes specify the length of the payload. Should not be used in Version 1.0
Message Payload (encrypted)	variable	No	See Table A.2
Message Integrity Check	variable	No	Contains an integrity check value. Must be present for all secured session types.

Table A.1: Matter Message Format

Name	Length	Required	Description
Exchange Flags	1 Byte	Yes	Consists of 5 subfields, each 1 Bit long (remaining 3 Bit are reserved for the future). Subfields (each 1 Bit long and in chronological order starting from position 0): I Flag (indicates if the message was sent from the exchange initiator), A Flag (indicates that the message shall be interpreted as an acknowledgment of a previous message), R Flag (indicates that the sender wishes for an acknowledgment for the message), SX Flag (indicates that the `Secured Extensions` section of the message is present; must be set to 0 for Version 1.0), V Flag (indicates that the `Protocol Vendor ID` section of the message is present).
Protocol Opcode	1 Byte	Yes	Unsigned integer value identifying the message's type.
Exchange ID	2 Bytes	Yes	Identifies the exchange the message belongs to. Represented by a unique unsigned integer value.
Protocol ID	2 Bytes	Yes	Unsigned integer value specifying in which protocol the `Protocol Opcode` is defined.
Protocol Vendor ID	2 Bytes	No	Unsigned integer value identifying the Vendor ID namespacing.
Acknowledged Message Counter	4 Bytes	No	Keeps track of the previous message that has been acknowledged by this message.
Secured Extensions	variable	No	Provides backwards compatible extensibility. The first 2 Bytes specify the length of the payload. Shall not be used in Version 1.0
Application Payload	variable	No	Contains the application data of the message.

Table A.2: Protocol Message Format

APPENDIX B

Thing Descriptons

B.1 Light Bulb

```
{
  "@context": "https://www.w3.org/2019/wot/td/v1",
  "title": "Matter Light Bulb",
  "id": "ff35:8388:2941:3b00:3d6f:ed16:b010:7607",
  "description": "Light Bulb Example",
  "securityDefinitions": {
    "psk_sc": {
      "scheme": "psk"
    }
  },
  "security": "psk_sc",
  "properties": {
    "OnOff": {
      "title": "OnOff",
      "observable": false,
      "readOnly": true,
      "type": "boolean",
      "forms": [
        {
          "op": [
            "readproperty"
          ],
          "href": "matter://ff35:8388:2941:3b00:3d6f:ed16:b010:7607/onoff/read/1/1",
          "matter:attributeName": "onoff"
        }
      ]
    },
    "CurrentLevel": {
      "title": "CurrentLevel",
      "observable": false,
      "readOnly": true,
      "minimum": 0,
      "maximum": 254,
      "type": "integer",
      "forms": [
```

```
        {
          "op": [
            "readproperty"
          ],
          "href": "matter://ff35:8388:2941:3b00:3d6f:ed16:b010:7607\
          /levelcontrol/read/1/1",
          "matter:attributeName": "currentlevel"
        }
      ]
    },
    "RemainingTime": {
      "title": "RemainingTime",
      "observable": false,
      "readOnly": true,
      "type": "integer",
      "forms": [
        {
          "op": [
            "readproperty"
          ],
          "href": "matter://ff35:8388:2941:3b00:3d6f:ed16:b010:7607\
          /levelcontrol/read/1/1",
          "matter:attributeName": "remainingtime"
        }
      ]
    },
    "OnOffTransitionTime": {
      "title": "OnOffTransitionTime",
      "observable": false,
      "readOnly": false,
      "type": "integer",
      "forms": [
        {
          "op": [
            "readproperty"
          ],
          "href": "matter://ff35:8388:2941:3b00:3d6f:ed16:b010:7607\
          /levelcontrol/read/1/1",
          "matter:attributeName": "onofftransitiontime"
        }
      ]
    },
    "DefaultMoveRate": {
      "title": "DefaultMoveRate",
      "observable": false,
      "readOnly": false,
      "type": "integer",
      "forms": [
        {
          "op": [
            "readproperty"
          ],
          "href": "matter://ff35:8388:2941:3b00:3d6f:ed16:b010:7607\
          /levelcontrol/read/1/1",
          "matter:attributeName": "defaultmoverate"
        }
      ]
    }
  },
  "actions": {
    "identify": {
```

```
      "title": "identify",
      "description": "Starts or stops the receiving device identifying itself.",
      "forms": [
        {
          "op": [
            "invokeaction"
          ],
          "href": "matter://ff35:8388:2941:3b00:3d6f:ed16:b010:7607\
          /identify/identify/1/0"
        }
      ]
    },
    "read": {
      "title": "read",
      "description": "Reads the value of the specified attribute",
      "forms": [
        {
          "op": [
            "invokeaction"
          ],
          "href": "matter://ff35:8388:2941:3b00:3d6f:ed16:b010:7607\
          /descriptor/read/1/0",
          "matter:attributeName": "attributeName"
        }
      ]
    },
    "write": {
      "title": "write",
      "description": "Assign the given value to the specified attribute",
      "forms": [
        {
          "op": [
            "invokeaction"
          ],
          "href": "matter://ff35:8388:2941:3b00:3d6f:ed16:b010:7607\
          /descriptor/read/1/0",
          "matter:attributeName": "attributeName",
          "matter:attributeValue": "attributeValue"
        }
      ]
    },
    "off": {
      "title": "off",
      "description": "set to Off",
      "forms": [
        {
          "op": [
            "invokeaction"
          ],
          "href": "matter://ff35:8388:2941:3b00:3d6f:ed16:b010:7607/onoff/off/1/1"
        }
      ]
    },
    "on": {
      "title": "on",
      "description": "set to On",
      "forms": [
        {
          "op": [
            "invokeaction"
          ],
```

```
158             "href": "matter://ff35:8388:2941:3b00:3d6f:ed16:b010:7607/onoff/on/1/1"
159           }
160         ]
161       },
162       "toggle": {
163         "title": "toggle",
164         "description": "toggle between On and Off",
165         "forms": [
166           {
167             "op": [
168               "invokeaction"
169             ],
170             "href": "matter://ff35:8388:2941:3b00:3d6f:ed16:b010:7607/onoff/toggle/1/1"
171           }
172         ]
173       },
174       "move-to-level": {
175         "title": "move-to-level",
176         "description": "transition to the desired brightness level",
177         "forms": [
178           {
179             "op": [
180               "invokeaction"
181             ],
182             "href": "matter://ff35:8388:2941:3b00:3d6f:ed16:b010:7607/onoff/off/1/1",
183             "matter:level": "desiredLevel",
184             "matter:transitionTime": "desiredTransitionTime"
185
186           }
187         ]
188       }
189     },
190     "events": {}
191   }
```

B.2 Light Switch

```
1   {
2     "@context": "https://www.w3.org/2019/wot/td/v1",
3     "title": "Matter Light Switch",
4     "id": "ff35:8388:2941:3b00:f5e2:c80e:94a2:6a8c",
5     "description": "Light Switch Example",
6     "securityDefinitions": {
7       "psk_sc": {
8         "scheme": "psk"
9       }
10    },
11    "security": "psk_sc",
12    "properties": {
13      "OnOff": {
14        "title": "OnOff",
15        "observable": false,
16        "readOnly": true,
17        "type": "boolean",
18        "forms": [
19          {
20            "op": [
```

```
            "readproperty"
          ],
          "href": "matter://ff35:8388:2941:3b00:f5e2:c80e:94a2:6a8c/onoff/read/2/1",
          "matter:attributeName": "onoff"
        }
      ]
    },
    "OnTime": {
      "title": "OnTime",
      "observable": false,
      "readOnly": false,
      "type": "integer",
      "forms": [
        {
          "op": [
            "readproperty"
          ],
          "href": "matter://ff35:8388:2941:3b00:f5e2:c80e:94a2:6a8c\
          /onoff/read/2/1",
          "matter:attributeName": "ontime"
        }
      ]
    }
  },
  "actions": {
    "identify": {
      "title": "identify",
      "description": "Starts or stops the receiving device identifying itself.",
      "forms": [
        {
          "op": [
            "invokeaction"
          ],
          "href": "matter://ff35:8388:2941:3b00:f5e2:c80e:94a2:6a8c\
          /identify/identify/2/0"
        }
      ]
    },
    "read": {
      "title": "read",
      "description": "Reads the value of the specified attribute",
      "forms": [
        {
          "op": [
            "invokeaction"
          ],
          "href": "matter://ff35:8388:2941:3b00:f5e2:c80e:94a2:6a8c\
          /descriptor/read/2/0",
          "matter:attributeName": "attributeName"
        }
      ]
    },
    "write": {
      "title": "write",
      "description": "Assign the given value to the specified attribute",
      "forms": [
        {
          "op": [
            "invokeaction"
          ],
          "href": "matter://ff35:8388:2941:3b00:f5e2:c80e:94a2:6a8c\
```

```
          /descriptor/write/2/0",
          "matter:attributeName": "attributeName",
          "matter:attributeValue": "attributeValue"
        }
      ]
    },
    "off": {
      "title": "off",
      "description": "set to Off",
      "forms": [
        {
          "op": [
            "invokeaction"
          ],
          "href": "matter://ff35:8388:2941:3b00:f5e2:c80e:94a2:6a8c/onoff/off/2/1"
        }
      ]
    },
    "on": {
      "title": "on",
      "description": "set to ON",
      "forms": [
        {
          "op": [
            "invokeaction"
          ],
          "href": "matter://ff35:8388:2941:3b00:f5e2:c80e:94a2:6a8c/onoff/on/2/1"
        }
      ]
    },
    "toggle": {
      "title": "toggle",
      "description": "toggle between On and Off",
      "forms": [
        {
          "op": [
            "invokeaction"
          ],
          "href": "matter://ff35:8388:2941:3b00:f5e2:c80e:94a2:6a8c/onoff/toggle/2/1"
        }
      ]
    }
  },
  "events": {}
}
```

List of Figures

List of Tables

Acronyms

ACL Access Control List. 41

AEAD Authenticated Encryption with Associated Data. 22, 23

APIs Application Programming Interfaces. 33

BLE Bluetooth Low Energy. 8–11, 24, 25, 40

BTP Bluetooth Transport Protocol. 16, 19

CASE Certificate Authenticated Session Establishment. 19, 21, 24

CD Certification Declaration. 27, 28

CMS Cryptographic Message Syntax. 27

DAC Device Attestation Certificate. 27

GAP Generic Access Profile. 25

GATT Generic Attribute Profile. 25

GUI Graphical User Interface. 39

HVAC Heating, Ventilation and Air Conditioning. 28

IoT Internet of Things. 11, 24, 33–35, 49

IP Internet Protocol. 12, 16, 17, 19, 24, 25

JSON-LD JavaScript Object Notation for Linked Data. 34, 35

MIC Message Integrity Check. 23

MRP Message Reliability Protocol. 21, 22

NOC Node Operational Certificate. 21, 28

OSI Open Systems Interconnection. 18

OTBR OpenThread Border Router. 38, 39

PAA Product Attestation Authority. 27

PAI Product Attestation Intermediate. 27

PASE Passwort-Authenticated Session Establishment. 19, 20, 24

PB Protocol Binding. 2–4, 37, 42, 43, 46–48

PBKDF Password-Based Key Derivation Function. 20

RCP Radio Co-Processor. 37–39

TCP Transmission Contol Protocol. 16, 19

TD Thing Description. 2–4, 33–37, 41, 42, 44, 46–48, 50

TLV Tag Length Value. 26, 27

UDP User Datagram Protocol. 16, 19, 21

URL Uniform Resource Locator. 42, 43, 47, 63

WoT Web of Things. 1–4, 33–35, 42–50

WWW World Wide Web. 33

Bibliography

[1] H. Strese, U. Seidel, T. Knape, and A. Botthof, *Smart Home in Deutschland*. Berlin, DE: Institut für Innovation und Technik Berlin, May 2010.

[2] C. Chilipirea, A. Ursache, D. O. Popa, and F. Pop, "Energy efficiency and robustness for IoT: Building a smart home security system," in *2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp. 43–48, Sept. 2016.

[3] Benjamin K. Sovacool and Dylan D. Furszyfer Del Rio, "Smart home technologies in Europe: A critical review of concepts, benefits, risks and policies | Elsevier Enhanced Reader," Mar. 2020.

[4] Y. Strengers and L. Nicholls, "Convenience and energy consumption in the smart home of the future: Industry visions from Australia and beyond," *Energy Research & Social Science*, vol. 32, pp. 86–93, Oct. 2017.

[5] A. Anvari-Moghaddam, H. Monsef, and A. Rahimi-Kian, "Optimal Smart Home Energy Management Considering Energy Saving and a Comfortable Lifestyle," *IEEE Transactions on Smart Grid*, vol. 6, pp. 324–332, Jan. 2015.

[6] N. Surantha and W. R. Wicaksono, "Design of Smart Home Security System using Object Recognition and PIR Sensor," *Procedia Computer Science*, vol. 135, pp. 465–472, Jan. 2018.

[7] X. Guo, Z. Shen, Y. Zhang, and T. Wu, "Review on the Application of Artificial Intelligence in Smart Homes," *Smart Cities*, vol. 2, pp. 402–420, Sept. 2019.

[8] S. S. I. Samuel, "A review of connectivity challenges in IoT-smart home," in *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, pp. 1–4, Mar. 2016.

[9] The Editors of Encyclopaedia Britannica, "Wi-Fi | Definition, Name, & Facts | Britannica." https://www.britannica.com/technology/Wi-Fi, Apr. 2023.

[10] Z. Ren, C. Chen, and L. Zhang, "Security Protection under the Environment of WiFi," in *2017 International Conference Advanced Engineering and Technology Research (AETR 2017)*, pp. 50–54, Atlantis Press, Mar. 2018.

[11] T. S. Andjamba, G.-A. L. Zodi, and D. S. Jat, "Interference analysis of IEEE 802.11 wireless networks: A case study of Namibia University of Science and Technology," in *2016 International Conference on ICT in Business Industry & Government (ICTBIG)*, pp. 1–5, Nov. 2016.

[12] P. Barker and M. Hammoudeh, "A Survey on Low Power Network Protocols for the Internet of Things and Wireless Sensor Networks," in *Proceedings of the International Conference on Future Networks and Distributed Systems*, (Cambridge United Kingdom), pp. 1–8, ACM, July 2017.

[13] C. Gomez, J. Oller, and J. Paradells, "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology," Tech. Rep. 9, Molecular Diversity Preservation International, Sept. 2012.

[14] G. Kaur, P. Tomar, and M. Tanque, *Artificial Intelligence to Solve Pervasive Internet of Things Issues*. Chennai: Mara Conner, 2021.

[15] K. Singh, "ZigBee: A Review," tech. rep., Jan. 2012.

[16] S. Khanji, F. Iqbal, and P. Hung, "ZigBee Security Vulnerabilities: Exploration and Evaluating," July 2019.

[17] I. Unwala, Z. Taqvi, and J. Lu, "Thread: An IoT Protocol," in *2018 IEEE Green Technologies Conference (GreenTech)*, pp. 161–167, Apr. 2018.

[18] W. Zegeye, A. Jemal, and K. Kornegay, "Connected Smart Home over Matter Protocol," in *2023 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 1–7, Jan. 2023.

[19] Document 22-27349, "Matter Specification Version 1.0," standard, Connectivity Standards Alliance, California, USA, Sept. 2022.

[20] H. Krawczyk, "SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols," in *Advances in Cryptology - CRYPTO 2003* (D. Boneh, ed.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 400–425, Springer, 2003.

[21] "The Device Data Model | Matter." https://developers.home.google.com/matter/primer/device-data-model.

[22] D. Guinard, *A Web of Things Application Architecture: Integrating the Real-World into the Web*. PhD thesis, ETH Zurich, 2011.

[23] "Web of Things (WoT) Architecture 1.1," standard, World Wide Web Consortium, Wakefield, USA, Nov. 2021.

[24] S. Kaebisch, M. McCool, and E. Korkan, "Web of Things (WoT) Thing Description 1.1," June 2023.

[25] Document 22-27350, “Matter Application Cluster Specification,” standard, Connectivity Standards Alliance, California, USA, Sept. 2022.