



universität ulm



Security Analysis of the Matter Protocol

Melissa Loos
XXXXXX

Master's Thesis
XXXXXXXXXXX

Examined by
Prof. Dr. rer. nat. Frank Kargl
Prof. Dr.-Ing. Franz Hauck

Supervised by
M.Sc. Stephan Kleber
M.Sc. Heiko Ehret

Institute of Distributed Systems
Faculty of Engineering, Computer Science and Psychology
Ulm University

January 31, 2023



© 2023 Melissa Loos

Issued: May 13, 2023



This work is licenced under a Creative Commons Attribution License.

To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

I hereby declare that this thesis titled:

Security Analysis of the Matter Protocol

is the product of my own independent work and that I have used no sources or materials other than those specified. The passages taken from other works, either verbatim or paraphrased in the spirit of the original quote, are identified in each individual case by indicating the source.
I further declare that all my academic work was written in line with the principles of proper academic research according to the official "Satzung der Universität Ulm zur Sicherung guter wissenschaftlicher Praxis" (University Statute for the Safeguarding of Proper Academic Practice).

Ulm, January 31, 2023

Melissa Loos, student number XXXXXX

ACKNOWLEDGEMENT

I want to thank all my co-workers at SCHUTZWERK, for all the funny and educational moments we share. A special thanks goes out to my supervisors Heiko Ehret (SCHUTZWERK) and Stephan Kleber (Ulm University), as well as Gabriele Mohr (SCHUTZWERK) for the linguistic QA. I also want to thank Prof. Dr. Frank Kargl, Prof. Dr. Hauck, Dr. Benjamin Erb, Dr. Christoph Bösch and Ala'a Al-Momani for the very informative and enabling lectures and lab sessions, that were exceptionally organized and thought-through. The well-structured lectures offered by you and your institute were a great introduction and immersion into the world of distributed systems, security, privacy and concurrency and I enjoyed your courses especially throughout the pandemic, because you managed to leverage the benefits of online lectures such that they stood not behind the lectures in presence. I also want to thank my former team at SGI that gave me a deep dive into the world of Linux, automation of distributed systems and system administration, that probably was the tipping point that shoved me towards security and aflamed my love for Linux, I am very grateful for what you taught me during my time in university. Lastly, I want to anonymously thank my love and all my friends for all the support and the beautiful moments we share, for the mutual trust and faith.

ABSTRACT

This thesis focuses on evaluating the security of the recently released home automation standard Matter, which is intended to improve the security and interoperability of smart home devices. The goal of this work was to analyze the security implications that can be derived from the specification of Matter, and to determine whether the Matter standard fulfills the promises made regarding its security. Our analysis was done by creating a threat model and performing a manual vulnerability analysis, where we focused on determining the attack surface and potential vulnerabilities, dangerous deviations that could be made by vendors, and by providing concise examples of possible implementations that could reduce the overall security of Matter in the field. In addition, we added a test catalog and Tamarin models of Matter's PAKE, SIGMA exchange during the CASE as well as plain SPAKE2+ for future use in symbolic verification. We were able to show that Matter is able to mostly fulfill its claims, explain where Matter deviates from its promises, and provide recommendations on how to further increase Matter's security. Our work also identifies weaknesses in the specification for which we expect future revisions of the specification to add additional mandatory requirements to address them. With this work, we aim to help security researchers, device manufacturers, and specification developers in gaining a comprehensive picture of Matter's security model, and ultimately contribute to overall security and privacy.

CONTENTS

Abstract v

Contents vii

1 Introduction 1
- 1.1 Related Work . . . 3
- 1.2 Research Methodology . . . 4
  - 1.2.1 Research Purpose . . . 4
  - 1.2.2 Research Questions . . . 4
  - 1.2.3 Research Methods . . . 5
- 1.3 Structure of the Thesis . . . 6

2 Theoretical Background 9
- 2.1 Protective Goals in IT Security . . . 9
  - 2.1.1 Confidentiality . . . 9
  - 2.1.2 Integrity . . . 9
  - 2.1.3 Availability . . . 9
  - 2.1.4 Authenticity . . . 10
  - 2.1.5 Non-Repudiation . . . 10
- 2.2 General Threats to Public Key Infrastructures . . . 10
- 2.3 Cryptographic Operations . . . 10
  - 2.3.1 Advanced Encryption Standard . . . 10
  - 2.3.2 SPAKE2+ . . . 12

3 Matter 15
- 3.1 Origin . . . 15
- 3.2 Goals . . . 16
- 3.3 Communication . . . 17
  - 3.3.1 Device Commissioning . . . 17
  - 3.3.2 Certificate Authenticated Session Establishment . . . 21
  - 3.3.3 Communication within a Fabric . . . 21
  - 3.3.4 Device Discovery . . . 23
  - 3.3.5 Matter over Thread . . . 24
  - 3.3.6 Sleepy End Device . . . 26
  - 3.3.7 Message Format . . . 26
  - 3.3.8 Bridging of Non-Matter IoT Devices . . . 26
- 3.4 Protocol Stack . . . 26
  - 3.4.1 Matter Data Model . . . 27
  - 3.4.2 Interaction Model . . . 28

3.5 Certificates . . . 28
3.5.1 Distributed Compliance Ledger . . . 28
3.5.2 Public Key Infrastructure . . . 29
3.6 Security Features . . . 30

4 Security Analysis **33**
4.1 Security Requirement Engineering . . . 33
4.1.1 Threat Modeling . . . 33
4.1.2 Development of Security Requirements . . . 34
4.1.3 Development of Security Mechanisms . . . 35
4.1.4 Threat Model . . . 35
4.1.5 Security and Privacy Requirements . . . 47
4.2 Review of the Matter Specification . . . 50
4.2.1 Cryptographic Primitives . . . 50
4.2.2 Security Analyis of the Cryptographic Primitives . . . 54
4.2.3 Use of Unsecured Communication . . . 60
4.2.4 Device Passcode and Onboarding Payload . . . 60
4.2.5 Current Time and Certificate Validation . . . 62
4.2.6 Security of Device Discovery . . . 63
4.2.7 Security of Device Commissioning . . . 64
4.2.8 URIs Provided by the DCL . . . 64
4.2.9 Underlying Communication Channels . . . 65
4.2.10 Multi-Admin and Multi-Fabric . . . 65
4.2.11 Security and Limitations of Identifiers . . . 66
4.2.12 Security of Sleepy End Devices . . . 67
4.2.13 Certificate Authenticated Session Establishments . . . 67
4.2.14 Message Counters . . . 67
4.2.15 Secure Channel Protocol . . . 68
4.2.16 Groupcast Communication . . . 68
4.2.17 Certificates and Key Material . . . 69
4.2.18 Access Control . . . 69
4.2.19 Data and Interaction Model . . . 70
4.2.20 Bridges . . . 70
4.2.21 Subscriptions and Proxies . . . 71
4.2.22 General Diagnostics Cluster . . . 71
4.2.23 Over-the-air Device-Firmware-Updates . . . 72
4.2.24 In-Field Upgrade to Matter . . . 73
4.2.25 Security Measures of Device Manufacturers and Recommended Countermeasures . . . 73
4.3 Symbolic Verification . . . 74

5 Discussion **75**
5.1 Remarkably Strong Cryptography . . . 75
5.1.1 Recommendation of Additional Acceptance Criteria for RNGs . . . 75

5.1.2 Hashes and HMACs Are Secure . . . 76
5.1.3 Matter Could Leverage the Full Advantages of AEAD . . . 76
5.1.4 Exceptionally Strong PAKE, Mix-Up of Versions . . . 76
5.2 Much of Matter's Security Depends on Passcode Secrecy . . . 78
5.3 The Usual Struggle with Time Synchronization . . . 82
5.4 Discovery is Inherently Interesting, Practical Investigations Recommended . . . 82
5.5 Commissioning – On Rogue Commissioners and Evil Twins . . . 84
5.5.1 The Rogue Commissioner Attack . . . 84
5.5.2 Device Attestation Verification Is Optional, Why Not Make It An Opt-Out? 85
5.5.3 The Evil Twin Attack . . . 85
5.5.4 Spoofing of the Pairing Hint Value . . . 86
5.5.5 Additional Thoughts on Commissioning . . . 87
5.6 Matter's Use of HTTPS-Based Uplinks . . . 87
5.7 Recommendation of Minimal Requirements for the Underlying Communication Channel . . . 88
5.8 Multi-Admin . . . 88
5.9 Identifiers . . . 91
5.10 Sleepy End Devices and Unsecured Parameter Exchanges . . . 92
5.11 Operational Session Establishment, Replay and Resumption . . . 92
5.12 Message Counters and Replay Attacks . . . 93
5.13 Status Report Message Injection, Bye-Bye Session Establishments . . . 94
5.14 Ambiguities in Groupcast Communication . . . 95
5.15 Epochs and Revocation . . . 95
5.16 Remarks on Counterfeiting . . . 97
5.17 On the Security of Certificates and Keys . . . 97
5.18 Access Control . . . 98
5.18.1 Administrative Privileges and Event Monitoring . . . 98
5.18.2 Group-Wide Permissions . . . 98
5.18.3 Bridges, Careful UI Design Required (Part I) . . . 99
5.19 Data and Interaction Model . . . 99
5.19.1 Timed Interactions, the Final Enemy of Replay Attacks . . . 99
5.19.2 Lists and Potential Memory Exhaustion . . . 100
5.19.3 Factory Resets . . . 100
5.19.4 Events and Network-Based Monitoring . . . 100
5.20 Bridges, Careful UI Design Required (Part II) . . . 101
5.21 Subscriptions and Proxies . . . 102
5.22 The General Diagnostics Cluster . . . 103
5.23 Security Considerations of Updates . . . 103
5.24 In-Field Upgrades to Matter . . . 104
5.25 Some Additional Thoughts . . . 105
5.26 Positive Remarks . . . 105
5.27 Future Research . . . 106
5.28 Limitations . . . 107

6 Conclusion **109**

Bibliography **113**

A Security Claims **121**

B Resources **127**

C Security Considerations Regarding BLE and Wi-Fi **131**

D Tamarin Theories **133**
Tamarin Theory: SPAKE2+ 133
Tamarin Theory: PAKE 143
Tamarin Theory: CASE 153

E Test Catalog **163**

F Attack Graphs **169**

List of Figures **180**

List of Tables **181**

1 INTRODUCTION

We are in a brave new world where not only do our children talk back to us, but so do our appliances.

— MOHAMMED DASTAGIR

Internet of Things (IoT) in its essence is the concept of interconnecting powered devices, extending the network or internet to device types that were not initially developed with interconnectivity in mind. Examples of such devices can be refrigerators, lighting in home appliances, but also industrially used devices such as heat sensors, implants and wind turbines. Many IoT ecosystems use a central platform to view data input and control the devices. Sometimes, these platforms offer analytics which can be used for automated decision-making, such as automatically turning on or off the heating to keep the temperature in a user-specified range, for which again such a platform might try to offer comfortable default values based on the analytics of usage patterns by the entirety of applicants, potentially regarding some sort of demographic data. Because of this, IoT offers its adopters a vast amount of benefits such as convenience and remote control of household devices that sometimes can work without human interference over long periods of time, which eventually convinced a great proportion of western population to adopt IoT. When living areas are equipped with networked technology, we speak of a *smart home*, a term used ambiguously to refer either to the equipped home itself or to one or more groups of networked IoT devices in a home. Jovanovic [49] claimed that in 2021, there were more than 10 billion active IoT devices, which would approximate a ratio of one active IoT device per human, while Thormundsson [85] reported that as of 2021, around 43 percent of US households responded to own a smart home device, increasing from 33 percent in 2019. This indicates a huge influence on everyday life for a great portion of people.

In research, IoT is a young and growing field [3] that brings a lot of chances, changes and challenges to home and industrial networks and IoT promises an inter-connectivity between computable objects that can transfer data with or without human interaction. Patel et al. [67] describe the following characteristics as fundamental to IoT: interconnectivity, things-related services, heterogeneity, dynamic changes, enormous scale, safety and connectivity. According to them, typical application areas are smart living (smart home appliances such as remotely-controlled lighting, safety monitoring such as remote smoke alarms, intrusion detection systems, energy and water use), smart cities (structural health, lightning, safety, transportation, smart parking, waste management), smart environments (air pollution monitoring, forest fire detection, weather monitoring, water quality, river floods, wildlife protection), smart industry (detection of gas levels and leakages, prediction of malfunctions and required service maintenance), smart health (patients surveillance, medical fridges, fall detection, dental health information, physical activity monitoring, implants), smart energy (smart grid, wind turbines and power houses, power supply controllers, photovoltaic installations) and smart agriculture (green houses, compost, animal farming and tracking, offspring care, field monitoring). The wide scope of appliance areas implies how there is scarcely any area that cannot profit from the use of IoT, and it is up to the reader to further investigate into each of these appliances.

However, interconnectivity comes as a double-edged sword, because it greatly extends the attack surface from the IT security perspective, requiring manufacturers and protocol engineers to carefully implement protective security measures. Securing a smart home is often a complex task, because typically, multiple low cost devices connect to a number of sensors, share data, and take control of each other. Sensitive information can be held or sent by home IoT devices, for example whether a door has just been locked or opened, or the password of a Wi-Fi network. Home IoT devices can be misused in a multitude of ways if not properly secured. If an attacker was, for example, able to access a security camera, he would be able to perform espionage against an individual or organization. Information disclosure, data tampering and damage to the devices must be prevented. For sensor networks where devices base decisions on sensor data, for example for health care IoT, additional safety threats are introduced, risking potential harm to the applicant, for example if the doses emitted by an insulin pump, or the sensor data on which the doses are calculated were modified. Another example would be modified traffic light, which may cause accidents and by, this harm bystanders, or could be abused for targeted crimes. Because of the great risk posed by various IoT appliances, combined with the great potential benefits of IoT, a great deal of attention must be paid to IoT security.

Furthermore, IoT devices can be misused for cyber attacks such as DDoS attacks like those that can be launched via Mirai-created botnets [5]; or their functionalities could be manipulated, so that, for example, the command to lock a door is not properly executed. It is to be expected that creators of botnets might target IoT devices at a great scale, which can result in significant damage at the receiving end of the attacks. Additionally, IoT devices can become the entry points to critical infrastructures if an IoT network was deployed, for example, at a power plant. Thus, considerable security analysis of the devices themselves as well as their communication protocols and the ecosystem as a whole is required to maintain security, safety and privacy during the use of these devices. In addition to these security concerns, interoperability between IoT devices from different manufacturers has been an ongoing issue until now, because typical devices would only interact within a vendor-specific ecosystem. Generally, standardized protocols are used to reduce problems related to interoperability and security, such as Bluetooth Low Energy (BLE). However, network and link layer protocols inherently do not contribute towards application layer interoperability, and thus, a new home automation standard by the name *Matter* was released, which defines the application layer and promises security-by-design, interoperability across devices of different manufacturers and to take user experience to the next level. Due to the significance of the companies involved in the organization behind Matter, the Connectivity Standards Alliance (CSA), it can be expected, that Matter is greatly welcomed and adopted by manufacturers and users in the following years. This thesis aims to analyze the security of Matter based on the initial specification of the standard, by shedding a light on technical aspects that could be or lead to vulnerabilities in the future, as well as underspecifications that could lead to dangerous deviations by manufacturers. By this, we want to contribute towards securing maximally interconnected IoT and both challenge and push further state-of-the-art IoT security.

To the best of the author's knowledge, there has not yet been any such thorough and independent analysis by independent academical research, and we therefore expect, that some of the results will lead to adjustments in future revisions of the Matter specification. By January 31, 2023, this thesis will be submitted to the CSA by digital means.

1.1 RELATED WORK

Since Matter is yet a very young IoT standard that has been released throughout the course of this thesis, few scientific literature has been released about it.

One of the network layer protocols used by Matter is *Thread*. Akestoridis et. al [2] analyzed the security of Thread networks by flashing OpenThread to development boards, analyzing traffic and studying energy depletion attacks as well as online password guessing attacks. Their work was motivated by Thread being the only IEEE 802.15.4-based networking protocol for Matter, and they were able to demonstrate how an outside attacker can cause energy depletion by preventing OpenThread devices to return to sleep mode. They noted that mitigating their fourth energy depletion attack was not simple because it leverages legitimately used unsecured IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) fragments. The researchers monitored the device power in watts, but unfortunately they did not quantify in terms of measuring or estimating for example devices, how the lifetime of a device's battery is reduced by the attack. This makes it harder to comprehend the impact, without reducing the overall contribution of their work. Furthermore, they presented how an attacker could guess the low-entropy password of a Thread device by using selective jamming and spoofing techniques. They rated the risk as low due to the minimum password length and rarity of new Thread devices entering the Thread network, which makes the online brute force harder to perform, but warned against hardcoding install codes and recommended to implement detection of consecutive incorrect password guesses to warn the user and suggested out-of-band password changing to allow users to protect against risks of potentially leaked passwords. Their contribution is valuable given that Thread is a young protocol whose security only few researchers have analyzed so far.

In July 2022, Salnikov [77] published a bachelor thesis in German language, dealing with the Matter protocol from a security perspective. In his work, a precisely specified version of one of the original reference applications of Matter was flashed on multiple micro controllers, for which he analyzed the network traffic. By testing a replay attack, he achieved to force a restart of one of the devices, but according to the thesis, the underlying error had been fixed throughout the course of the thesis. He also successfully performed a de-authentication attack, showing that Matter is not able to protect devices against vulnerabilities of the underlying Wi-Fi connection below, breaking the promise of Matter to protect against vulnerabilities of the technologies it is based on. He concludes that the current implementation of Matter offers a high amount of security, but still exhibits security flaws in its implementation in its pre-release state. Note that the implementation-first specification of Matter 1.0 was not available in July 2022, and therefore, it is unclear whether the security flaws resulted from an implementation flaw or the specification itself. Unfortunately, Salnikov's security analysis does not follow a formal, systematic approach.

Apart from that, various organizations in the Matter Working Group have published content about Matter. Apart from the high-level overview of Matter the CSA offers on their website, they have published a white paper on the security and privacy of the protocol in March 2022 [29]. Infineon has released a white paper about Matter in 2022 [47], in which they explain some technical details about the protocol from a high-level perspective, such as the protocol stack, device commissioning and security measures. Another master thesis on Matter has been published by Ganassin [38] in Italian language, which dealt with the analysis and implementation of Matter, re-

sulting in a new firmware module implementing the Matter protocol for a company, but the access to it is restricted by the associated university. Due to the potentially low relevance of Ganassin's thesis, we did not request access to the thesis.

On October 4, 2022, Version 1.0 of the Matter specification has been published [28], which focuses on outlining fundamental requirements of the protocol flow, data model and cryptographic primitives, partially pointing to other references that additionally apply. Even though Matter follows an implementation-first approach, the specification was given precedence in case of discrepancies, making it the most authoritative description of how Matter is supposed to work. In Chapter 3, an in-depth, but significantly shorter, mainly technical description of Matter is provided, necessary for readers that are not familiar with the specification to follow the course of this thesis.

1.2 RESEARCH METHODOLOGY

The next three subsections describe the research purpose, research questions and research methods.

1.2.1 RESEARCH PURPOSE

The goal of this research is to assess the security of the Matter standard to determine weaknesses and potential security flaws that potentially require further attention towards developing a specification for securely interconnected IoT communication across a large range of manufacturers. From this, future research questions can be derived to further assess the security of the Matter standard in an academic setting.

1.2.2 RESEARCH QUESTIONS

The research questions underlying this work are:

How secure is Matter?

1. How can the security of Matter be assessed using a systematic approach?
2. What are the results when performing the systematic analysis approach on Matter?
3. What dangerous deviations to the Matter specification could be introduced by device manufacturers?
4. What types of implementations within the standard could reduce the security of Matter in the field?

1.2.3 RESEARCH METHODS

The research was conducted as follows: Since Matter was not yet released at the beginning while writing this thesis, an Internet was searched for scientific literature as well as blog posts and other online content about Matter by conducting online searches using the terms 'Matter iot csa' and 'project chip iot csa' (without quotation marks) on Google Scholar (sorted by both relevance and year), Google, DuckDuckGo and YouTube. The scientific literature found is described in Section 1.1, and most of the sources found during the research have contributed towards writing together the theoretical background in Chapter 2, which, among others, provided an overview of the Matter specification and some technical insights, which are now presented in chapter 3. The theoretical background has been supplemented with a short description of protective goals in IT security and general threats to a Public Key Infrastructure (PKI).

Prior to Matter's release on October 4, 2022, most parts of the chapter on Matter were written up based on manufacturer documentation found online as cited, blog posts and video material released by various members of the CSA. Access to the specification prior to release was requested and not granted. After release of the specification, the chapter was finalized and checked for correctness, and, where necessary, adjusted based on the information directly extracted from the specification. Several approaches to assess the security of Matter have been considered by conducting a search for academic and sector specific literature, among them:

- Threat Vulnerability and Risk Assessment (TVRA)
    - Threat Analysis
    - Vulnerability Analysis
    - Risk Analysis
- Attack Surface Analysis
- Symbolic Security Protocol Verification
- Computational Security Protocol Verification
- Threat Modeling
- Attack Graphs
- Threat Trees
- Abuse Case Graphs
- Data Flow Diagrams (DFDs)
- DREAD
- STRIDE
- CIA
- LINDDUN
- Common Criteria (CC)
- Requirement Engineering

- Risk Management
- Penetration Testing
- The classical approach: 'many-eyes-principle' (unsystematic and potentially uncoordinated review by many security experts)

Note that the list mixes high-level and detail-level concepts, which may overlap, for the purpose of giving a complete overview of the available options. Of these options, a mixture of threat analysis, vulnerability analysis, risk analysis, threat modeling, attack graphs, DFDs, STRIDE, CIA, LINDDUN, requirement engineering the classical approach was chosen to manage the trade-off between time and reproducibility, following guidance on threat modeling by Shostack [78], Open Web Application Security Project (OWASP) [22] and Myagmar [57]. Additionally, symbolic security protocol verification was chosen, because it promises a fully reproducible and potentially scalable solution to address the asymmetry of work defensive security requires when compared to the attacker side. For this, working Tamarin theories have been created for both the original SPAKE2+ and Matter's Password Authenticated Session Establishment (PASE), which uses a variant of SPAKE2+ mixing several versions of the draft, as well as an incomplete draft of Matter's SIGn-and-MAc (SIGMA) exchange during the Certificate Authenticated Session Establishment (CASE). By this, we contribute towards the symbolic verification of the protocols used in Matter which may be used to base further research on, given that the theories lack of terminating security lemmata. The requirement engineering approach was discarded, after a large amount of security claims made by the developers of Matter were found. Instead, the analysis will be used to support, counter or limit each of these claims, see Section 3.6 for a summary of the security and privacy claims. This seemed to make more sense after we realized that Matter's wide area of application made it practically almost impossible to narrow out any potential security requirements. Hence, we decided to focus only on the claims that were provided by Matter, acknowledging that the standard is not required to fulfill more than it has promised, while keeping the freedom to object if we find an important security feature is missing. CIA, STRIDE and LINDDUN were chosen over DREAD due to their support in OWASP Threat Dragon [40], a free tool for creating DFDs for threat modeling, and because they are commonly used methods in IT security. The attack graphs and the DFD of Matter's PASE as well as the threat model were drafted prior to the release of the Matter specification. The second half of the thesis, right after the specification was released, was based on the classical approach, which is inherently less systematic, but promising in yielding significant results when used on young technologies and specifications that yet have to prove themselves and have not yet been openly analyzed. Note that there inherently is no deterministic approach due to the high workload of security analyses that is caused by the undecidability of the decision of secure versus unsecure for all protocols that have not already been proven unsecure. We submitted this thesis to the CSA to share our thoughts with them and allow them to be the first to gain knowledge of the potential weaknesses we discuss in later chapters.

1.3 STRUCTURE OF THE THESIS

The report continues in Chapter 2 with a brief summary of the theoretical background necessary to understand the remainder of the thesis, and continues in Chapter 3 with an introduction into

Matter, including the features we found most relevant. In Chapter 4, we provide a threat model for Matter and a security review of the specification, in which we point out essential security features of Matter. In Chapter 5 we discuss Matter's security features and assess in how far Matter adheres to our requirements and fulfills its claims, briefly describe the limitations of our work and add recommendations for future research. The report ends in Chapter 6 by summarizing our work. A list of figures, tables and acronyms as well as appendices are provided at the end of the thesis.

2 THEORETICAL BACKGROUND

> *When we talk about the Internet of Things, it's not just putting RFID tags on some dumb thing so we smart people know where that dumb thing is. It's about embedding intelligence so things become smarter and do more than they were proposed to do.*
>
> — NICHOLAS NEGROPONTE

In this section we briefly outline protective goals in IT security and general threats to Public Key Infrastructure (PKI) and provide general details on Advanced Encryption Standard (AES) and SPAKE2+, two cryptographic algorithms used in Matter.

2.1 PROTECTIVE GOALS IN IT SECURITY

During the course of this thesis, we consider the following five protective goals described below: Confidentiality, Integrity, Availability, Authenticity and Non-Repudiation.

2.1.1 CONFIDENTIALITY

Confidentiality describes the protection of information against unauthorized access. In the context of Matter, we are especially interested in the information in transit as well as network or other sensitive information stored on devices as a result of their association with a Matter network. Confidentiality is often protected by using encryption.

2.1.2 INTEGRITY

Integrity refers to the state where data remains unchanged by any unauthorized manipulation. In the context of Matter, we are especially interested in the integrity of data in transit between two or more devices within a Matter network. Integrity is often protected by using a checksum or non-malleable ciphers.

2.1.3 AVAILABILITY

Availability refers to the availability of data and functions in a system. In the context of security, the compromise of availability causes a device to be a lower ratio of time available when compared to the overall time ratio regarding availability of an uncompromised state. Availability is often protected by implementing Denial-of-Service (DoS) attack mitigation and fail state recovery.

2.1.4 AUTHENTICITY

By authenticity we understand the validity of a subject that is authorized to access a specific asset, which is established using a unique identity and characteristic traits associated with this identity. Authenticity is often protected by certificates (e.g., X.509 public key certificates).

2.1.5 NON-REPUDIATION

Non-repudiation means that a subject is not capable of denying the origin, authenticity and integrity of a message. Non-repudiation is often implemented using digital signatures.

2.2 GENERAL THREATS TO PUBLIC KEY INFRASTRUCTURES

General threats to PKI include, but are not limited to disclosure or loss of a private key, key collision, inappropriate signing of a certificate from an unauthorized source, weak Random Number Generators (RNGs) used for the creation of keys, disclosure of the private Certification Authority (CA) key and signing of certificates the CA is not authorized to sign. When a Distributed Compliance Ledger (DCL) is used to store and publish certificates and public keys, a DoS on the DCL can cause a DoS of the commissioning into home automation systems and introduce a potential trade-off between robustness and security.

2.3 CRYPTOGRAPHIC OPERATIONS

Below is a description of two of the cryptographic protocols used in Matter, including a brief description of publicly known vulnerabilities:

2.3.1 ADVANCED ENCRYPTION STANDARD

Advanced Encryption Standard (AES) is a Federal Information Processing Standards (FIPS)-approved cryptographic algorithm that can be used in different modes of which Matter uses Counter with CBC-MAC (CCM) as Authenticated Encryption with Associated Data (AEAD) mode, and Cipher Block Chaining (CBC) and Counter (CTR) as confidentiality-only modes.

Matter uses AES with CCM and 128-bit encryption keys (short: AES-CCM-128), a confidentiality-only CBC mode (see Figure 2.1) combined with a Message Authentication Code (MAC) which adds integrity protection and authentication during data transfer, and CTR mode (see Figure 2.2). CCM is defined in RFC 3610 [93]. Note that AES does not provide non-repudiation [1]. CCM is used in an *authenticate-then-encrypt* manner, where CBC-MAC is used to compute a MAC over the plain text, by calculating CBC over the message with a zero Initialization Vector (IV). Subsequently, CTR mode is used to encrypt both plain text and tag. The same encryption key is used for both modes.

1 Non-repudiation is commonly achieved by using digital signatures, which are out-of-scope for AES.

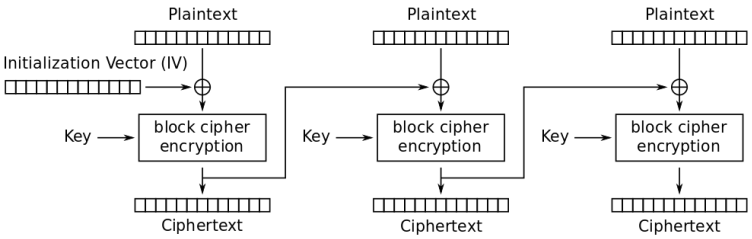


Figure 2.1: Cipher Block Chaining mode encryption. The image is unmodified and in the public domain. Source: Wikimedia Commons [91]

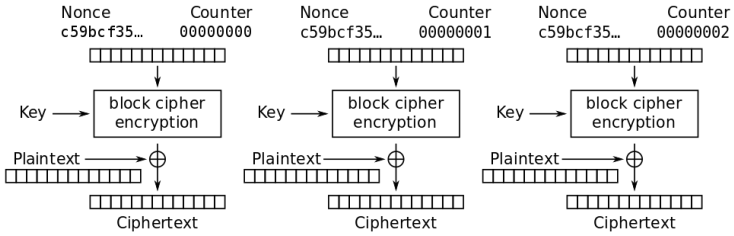


Figure 2.2: Counter mode encryption. The image is unmodified and in the public domain. Source: Wikimedia Commons [92]

Because CCM uses CTR in streaming mode (although AES in general is a block cipher), it is potentially vulnerable against attacks that are used against stream ciphers, including (1) bit-flipping attacks, (2) reused key attacks and (3) chosen-IV attacks.

The *reused key attack* can be described by the following formula:

$E(A) \oplus E(B) = (A \oplus C) \oplus (B \oplus C) = A \oplus B \oplus C \oplus C = A \oplus B$

It allows to recover plain texts if two distinct messages are encrypted using the same combination of key and nonce. Because of its relation to the *birthday problem* these attacks are sometimes also referred to as *birthday attacks*, and devices that encrypt and send a high number of messages are more susceptible to this attack. The attack can be mitigated by using an IV, sometimes also referred to as *nonce*, that may be sent in the clear, to create a one-time key for the stream cipher. However, this merely increases the entropy of the encryption key, thus decreases the chance of key collision, but does not fully prevent the attack. In practice, increasing the key and nonce entropy to a level where attacks are highly unlikely to occur results in *computational security* [2], even when *information-theoretic security* [3] is impossible to achieve.

2 We consider a computationally secure system secure in practice, because it can be assumed that no attacker will have a sufficient combination of time and resources required for cryptanalysis, to break the system.

3 Information-theoretically secure systems are secure against adversaries independent of their time and resources. Other

The *bit-flipping attack* sometimes occurs in the context of Machine-in-the-Middle (MitM) or replay attacks. It allows the attacker to modify the message's plain text without knowing the encryption key. While for messages with random plain text (or multi-layer encryption), the attack will result in garbage output, the attack can be very successful when used against known or predictable plain text, especially if the attacker knows the offset of a specific value, such as an Internet Protocol (IP) address, as was demonstrated by Rupprecht et al. in 2019 in the *aLTEr attack* [75]. The bit-flipping attack is mitigated using a MAC, which increases the likelihood that malicious messages are detected. Since CCM uses both MAC and IV, both reused key attacks and bit-flipping attacks are mitigated, but not fully prevented.

The third known attack to stream ciphers is the chosen-IV attack. In this attack, the attacker chooses specific IVs in a way that they reveal information about the secret key. The attack can be successful if chosen IVs produce non-random patterns of cipher text, which the attacker can use to narrow down the set of possible secret keys using differential cryptanalysis, and to infer information about the plain text. Therefore, this attack practically reduces the key entropy [4], and therefore the security provided by the key length. Note that for CCM, the IV used in CBC-MAC must not collide with the counter values used by the CTR mode. Ronen et al. demonstrated in 2018, how a side channel attack could be used to extract the AES-CCM key, which they believed could be used to jam Wi-Fi communication in a whole city, attack the electrical grid by toggling on and off light bulbs, or cause epileptic seizures by making a device flash at the right frequency [5] [73].

2.3.2 SPAKE2+

During the Password Authenticated Session Establishment (PASE), the SPAKE2+ [15] [82] protocol is used to establish a shared secret for symmetric encryption, for which one party uses a password and the other party uses a verification value, which can be computed during *offline initialization*. SPAKE2+ is an asymmetric (augmented) Password Authenticated Key Exchange (PAKE) protocol, in contrast to SPAKE2, which is a symmetric PAKE protocol, meaning that for SPAKE2, both parties have to know the password, where it is assumed that the server stores the password in a password file, whereas the password can be memorized by the user at the client side and is therefore not assumed to be stored [79]. In the context of Matter, the client side is taken by the commissioner (controller), and the server side is taken by the commissionee (joining device). The protocol requires at least two messages from the client side, and one from the server side [6]. Therefore, the protocol can be considered a two-round protocol. See Figure 2.3 for a depiction of the protocol flow. The protocol relies on the hardness of the Computational Diffie–Hellman (CDH) problem.

Before the start of the first phase, a preamble phase, also referred to as offline initialization, occurs during which certain information such as the identity and PAKE parameters are exchanged. More precisely, two values w0 and w1 are derived by hashing the password with the identities of

than many computationally secure algorithms, information-theoretic algorithms are considered quantum-safe.

4 Entropy is often misunderstood as a the key length. However, the entropy should rather be considered the bits of information unknown to the attacker.

5 Note that the attack demonstrates the importance of protecting even those devices which are usually not considered security-critical, because attacks can target critical infrastructure at a larger scale.

6 The exchanged values are symmetric, but the server can combine both messages to send one instead.

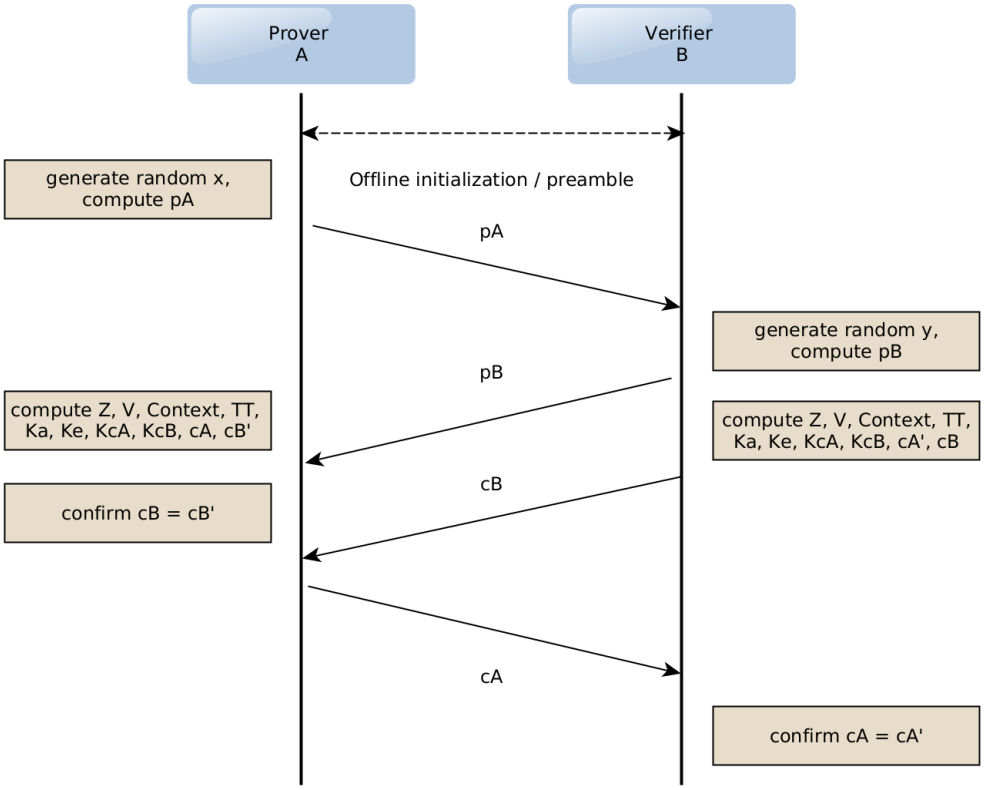


Figure 2.3: Protocol flow of SPAKE2+. Based on draft version 2 of SPAKE2+ [82]

prover A and verifier B. A stores $(\mathrm{w0}, \mathrm{w1})$, while B stores $(\mathrm{w0}, L = \mathrm{w1} \cdot p)$. The specification of SPAKE2+ makes clear, that protocols using SPAKE2+ must define the method used to compute w0 and w1. The method should ensure that w0 and w1 are indistinguishable from two uniformly random elements in the range $[0, p - 1]$. The draft recommends to use a Password-Based Key Derivation Function (PBKDF) to generate w0 and w1, for example *Scrypt* or *Argon2id*, to slow down brute-force attacks. To remove statistical bias by modular reduction, the output length of the PBKDF must be at least 64 bits longer than p, a large prime used as modulo for the elliptic curve group. It is recommended to compute w0 and w1 as follows:

$$\begin{aligned} \mathrm{w0s} \parallel \mathrm{w1s} &= \mathrm{PBKDF}(\mathrm{length}(\mathrm{pw}) \parallel \mathrm{pw} \parallel \mathrm{length}(A) \parallel A \parallel \mathrm{length}(B) \parallel B) \\ \mathrm{w0} &= \mathrm{w0s} \mod p \\ \mathrm{w1} &= \mathrm{w1s} \mod p \end{aligned}$$

During the first round, A and B exchange their public shares pA and pB. From these, A and B derive a shared secret which is used to derive the keying material used for encryption and authentication. During the second round, A and B calculate common values Z and V, the Context and protocol

transcript TT, the shared symmetric keys Ka, Ke, the key confirmation values KcA, KcB from which cA and cB are derived, and then exchange their distinct key confirmation messages cA and cB:

$$\begin{aligned} \mathrm{Ka} \parallel \mathrm{Ke} &= \mathrm{hash}(\mathrm{TT}) \\ \mathrm{KcA} \parallel \mathrm{KcB} &= \mathrm{KDF}(\mathrm{nil}, \mathrm{Ka}, \text{“ConfirmationKeys”}) \\ \mathrm{cA} &= \mathrm{mac}(\mathrm{KcA}, \ldots) \\ \mathrm{cB} &= \mathrm{mac}(\mathrm{KcB}, \ldots) \end{aligned}$$

According to the SPAKE2+ draft [82], identity information should be non-empty to avoid Unknown Key Share (UKS) attacks [11]. Furthermore, it must be checked that the elements received from a peer are in the prime order subgroup of G by co-factor multiplication and group members, as failure of these checks may introduce attack surface. It is important that random numbers are chosen truly uniform and are not reused as this would allow dictionary attacks on the password [83]. In the following chapter, we continue by providing a brief overview of Matter.

3 MATTER

> *We have a deep need and desire to connect. Everything in the history of communication technology suggests we will take advantage of every opportunity to connect more richly and deeply. I see no evidence for a reversal of that trend.*
>
> — PETER MORVILLE

This chapter outlines the core concepts and technical aspects of Matter and relevant background information. The chapter starts with an overview of the history of Matter and its goal and continues with describing various relevant technical aspects such as communication, then goes on with an enlistment of security features and cryptographic operations used in Matter. Information that is not relevant with regard to the subsequent security analysis is mostly omitted.

3.1 ORIGIN

Matter is a royalty-free home automation connectivity standard that was formerly known as Project Connected Home over IP (CHIP) which is guided by the Connectivity Standards Alliance (CSA), formerly known as Zigbee Alliance, and builds upon proven technologies. It is an application layer standard aiming to unify the smart home industry, and promises reliable and secure connectivity where devices work seamlessly together. The Matter Working Group was officially opened on January 17, 2020 [27] and published Version 1.0 of the specification on October 4, 2022 [28].

The vision of Matter is to unify the market and address interoperability, choice and ease of use, enable faster, easier and less costly development of IoT products, accelerate innovation, make retail easier and improve costumer purchase experience while enabling greater and accelerated adoption, add user control over device authorization and address security by leveraging modern security practices and protocols, using a federated approach where no single entity serves as a throttle or a single-point-of-failure for root of trust. Furthermore, Matter is meant to be robust and shall be implementable with low computational overhead [27] [54].

Currently, there is a high fragmentation of the smart home market, which lacks interoperability between ecosystems of different manufacturers. As a result, user choice is limited, which can degrade their experience. Having many different ecosystems using non-interoperable protocols leads to higher development costs because developers have to implement each ecosystem's protocols and implement very similar variants of each, to the confusion of users. Another problem is that some home IoT devices depend on cloud connectivity and are rendered useless whenever the company producing them closes the services, but since all Matter devices are required to be able to operate cloud-independent, future proofing is added to the world of home IoT [46].

Matter is supposed to solve these problems [54]. Wolf et al. thoroughly discussed the impact of Matter as an example of coopetition, or cooperative competition, on the home IoT market in [96]. As of now, more than 230 companies and 2300 individual members participate in the project, including big tech players like Google, Apple, Amazon and Samsung [26]. The connectivity standard

follows an open source and implementation-first development approach [27]. The specification is publicly available [28], but the availability of versions in development is limited to project members for an annual fee of 20,000 USD [1]. The open source implementation is available on GitHub, including reference applications such as a light bulb and mobile application [27]. The implementation will be ported to Portable Operating System Interface (POSIX) platforms (e.g., Linux) as well as embedded platforms for Integrated Circuit (IC) vendors [54]. Because Matter uses a public GitHub repository, security experts and attacker alike might retrieve information about bugs and non-standard-compliant behavior from it to identify vulnerabilities of the reference implementations or specification. Therefore, it seems reasonable to keep an eye on the issues marked with the *security* label to mitigate accidental or purposeful non-responsible disclosure.

3.2 GOALS

With Matter, the project members aim to define, develop and deliver a comprehensive application layer and data model which enables communication across smart home devices and mobile apps. Other than, for example, Bluetooth mesh or Zigbee, it is not full-stack, but instead reuses contributions from proven technologies by running on top of layers of other connectivity technologies and by inheriting their data models, which are used to organize data elements [27]. On the network layer, the standard runs on Internet Protocol version 6 (IPv6), and on the transport layer on Transmission Control Protocol (TCP) or User Datagram Protocol (UDP), which are protocols used for the mainstream internet and therefore allow extension to the Internet and cloud, although any outside connections are optional and any Matter device can operate entirely on a local basis [54]. Prior to using IP, non-IP mesh networks were required to adapt network headers, addresses and payload security to IP, which was often done by proprietary translation, potentially weakening end-to-end security. Furthermore, using IP, developers can use the same tools that they use for other internet-related devices and therefore development is sped up [54]. At launch, Matter will run on top of Thread, Wi-Fi and Ethernet [27], which are complementary because Thread runs on low power, low bandwidth devices (e.g. battery-operated devices such as sensors or door locks), while Wi-Fi can be used for high bandwidth and low latency transmissions by devices with a larger energy budget (e.g. security cameras). One goal of the Matter standard is to provide support for low energy nodes [30, Chapter 2.9]. For device commissioning, Bluetooth Low Energy (BLE) is used [27]. Further IP-capable network protocols may be supported in the future.

A fundamental design tenet of Matter lies on security [27]. Its security architecture is based on best practices such as *security by design* and *zero trust*. Therefore, every Matter device joining the network is authenticated and every message is protected by encryption, using proven cryptographic algorithms with high entropy. Furthermore, Matter requires support for secure Over-the-air Device Firmware Updates (OTA DFUs) for every certificated Matter device [46]. Every Matter product has to undergo a certification program to ensure it conforms to the standard specifications [25]. It can be assumed that for each product only a fixed amount of devices will be tested due to the impossibility of testing every single device that is being shipped.

1 CSA Subscription Plan. https://csa-iot.org/become-member/ [Last accessed: 2022-08-11]

3.3 COMMUNICATION

In the following sections, a brief overview of the communication phases of Matter is provided. These include the device commissioning, discovery and subsequent session establishments. Figure 3.1 shows the Matter protocol within the context of the IP stack.

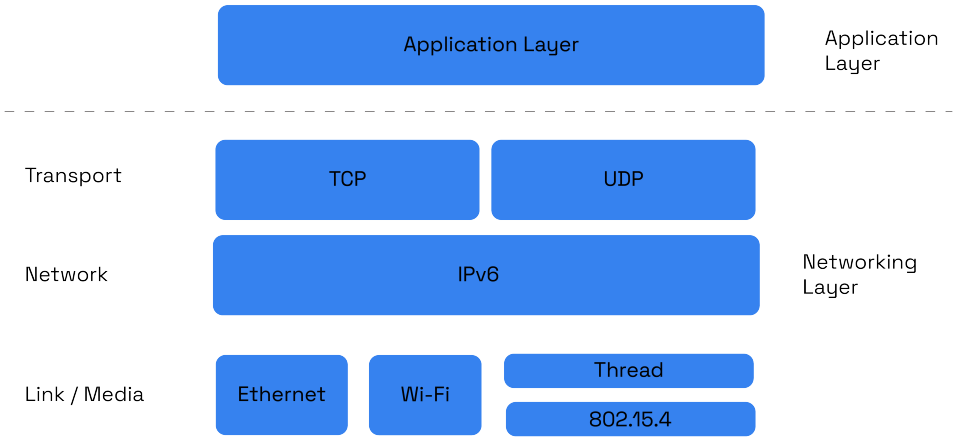


Figure 3.1: Matter defines the application layer within the IP stack. The image is unmodified and licensed under Apache License 2.0. Source: Matter code repository [27]

3.3.1 DEVICE COMMISSIONING

Whenever a new device wants to connect to a Matter network, commissioning must be performed by a Matter controller (e.g., a mobile device), called commissioner [7] [54]. Commissioning is used to authenticate the device, set up the network and install the Node Operational Certificate (NOC) on the device [48]. The authentication is performed by a handshake with the device in which the Certification Declaration (CD) is obtained, the signature verified, a copy of the Device Attestation Certificate (DAC) is obtained, and its link to CSA verified, confirming that this specific device is authentic [47]. Device commissioning is performed over IP if the device is already in an IP network, otherwise it is performed over BLE. BLE commissioning (discovery and connection establishment) is done over Generic Access Profile (GAP) and credentials are transmitted over Bluetooth Generic Attribute Profile (GATT). During Wi-Fi commissioning, the commissioner acts as a Soft-AP. Standard Wi-Fi AP advertisements are used for device discovery [30, Chapter 5.4.1].

The specification non-exhaustively describes four ways to commission a device: (1) the setup of a not yet commissioned device with either commissioner setup code entry or (2) user-initiated beacon detection, (3) the setup of an already commissioned device with user-initiated beacon detection, and (4) the setup of an On-Network device using commissioner discovery [30, Chapter

5.2.1]. There are two commissioning flows depending on whether the device is capable of concurrent network connections, or not. If the device is capable of concurrent network connections, the commissionee and commissioner are connected with each other over the commissioning channel and to the operational network [30, Chapter 5.5].

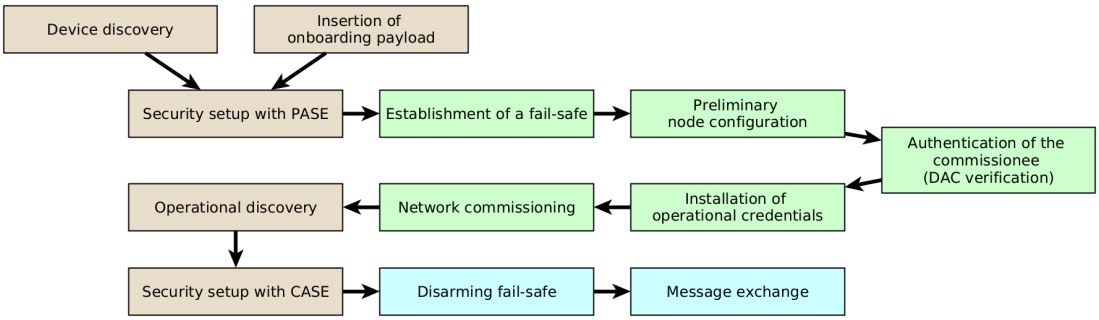


Figure 3.2: Overview of the commissioning phases. Brown depicts unsecured communication, green is PASE-secured and blue is CASE-secured. Modified version of Nordic Semiconductor's depiction [64]

The commissioning process for a not-yet commissioned device is started by entering the onboarding payload into the commissioner, which is unique to a device and establishes its identity in form of the *device discriminator* [65]. The onboarding payload also includes a setup code and cryptographic information required for the secure connection [7] [54]. Onboarding payload can come in different forms to support as many different devices as possible. The specification describes onboarding payload in written form, as a Quick Response Code (QR code) or Near Field Communication (NFC) tag [62]. Entering the setup code into the commissioner will start the *device discovery* stage. By pushing the pairing button on the device, the new device starts beaconing that it needs to be added to the network and advertises its identity information. The commissioner then discovers the commissionee by listening to the announcements on the network and looking out for the device, whose code was inserted [47]. Commissioning can be attempted with multiple devices depending on the result of the discovery [30, Chapter 5.2.2.1]. The next stage is the *security setup with PASE* (see Section 3.3.1), during which commissioner and commissionee establish a secure connection using the setup code as the Out-of-band (OOB)-passcode to derive two unidirectional encryption keys along with an attestation challenge [48] [30, Chapter 2.8]. This passcode should only be known to the commissioner and commissionee [65]. Commissionees may also initiate the commissioning process by discovering commissioners [30, Chapter 4.3.3]. Once the secure initial session is established, the commissioner will ask the commissionee to back up its original configuration to *establish a fail-safe*. The commissioner will then proceed with *preliminary node configuration* during which it configures the commissionee with regulatory information and the current Coordinated Universal Time (UTC) time [65]. The next stage is the *DAC verification*, during which the commissioner verifies whether the commissionee is a certified Matter device. For the verification to succeed, the commissionee must prove ownership and validity of mandatory device attestation elements [54] [65]. To achieve this, the commissioner will request the commissionee's DAC [48] [54] and notify the user if the device is not certified [30, Chapter 2.8]. Once the

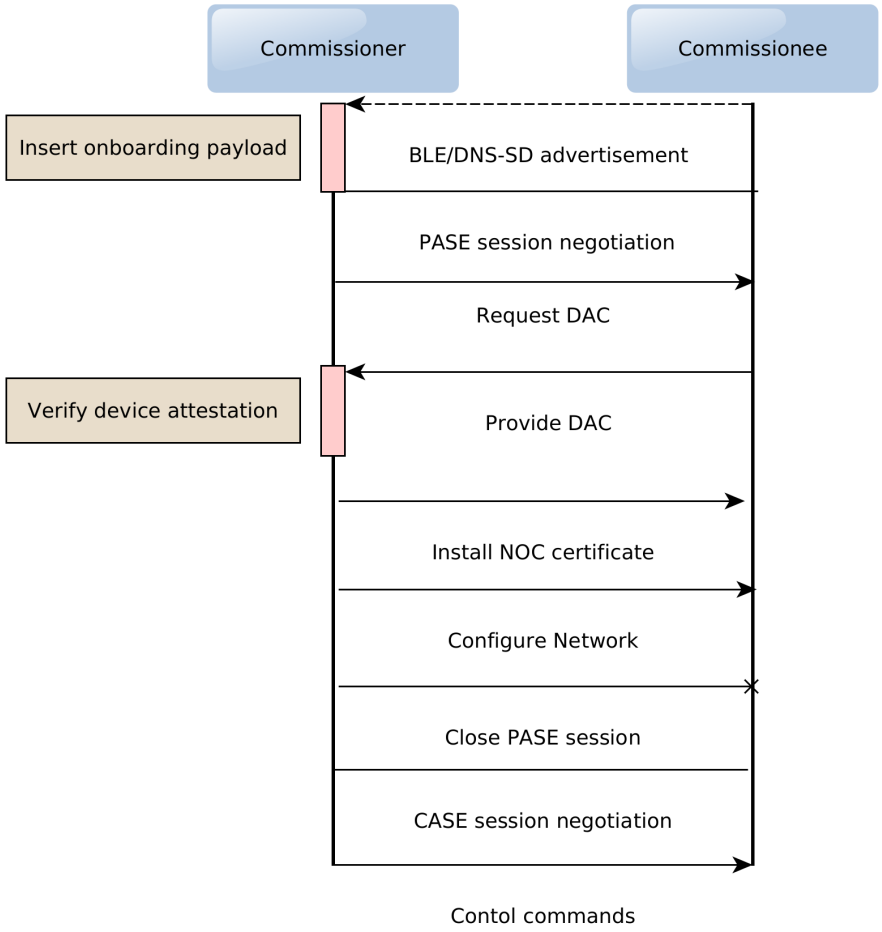


Figure 3.3: Overview of the initial session establishment

commissionee is verified, the commissioner starts provisioning it [47] during the *information configuration* stage, by *installing operational credentials* in form of a NOC and an operational ID on the commissionee [48] [65] and providing information on the regulatory domain, UTC time and network interfaces configuration [30, Chapter 2.8]. The NOC can either be signed by an Intermediate Certification Authority (ICA) or by the Root Certificate Authority (RCA), the latter of which is directly generated by the commissioner [48]. By this, the commissionee becomes a new node in the ecosystem, which is a Matter network called *fabric* in Matter terminology. Fabrics are identified using the two-element tuple `(Root PK, Fabric ID)` [65]. Using *multi-admin*, a foundational feature in Matter, users can connect `n` Matter devices to `m` applications and `o` fabrics locally and simultaneously [54] [65], which was difficult or impossible prior to Matter.

The commissioning network is distinct from the operational network to which the device is added; in order for a device to join the operational network, it must obtain the network name (e.g., Service Set Identifier (SSID)) and network key (e.g., Wi-Fi password) unless it was already con-

nected prior to commissioning. The commissioner either already knows the device's IPv6 address or discovers it during this stage [30, Chapter 2.8]. This is done during the *network commissioning* stage in which the commissioner sends all necessary information that are required to join the operational network to the commissionee [47] [48] [65]. Once the joining device is correctly configured, the session is closed [48] during the *non-concurrent connection commissioning flow*, while the *concurrent connection commissioning flow* allows the nodes to be connected to the operational network and the commissioning channel simultaneously [65]. Once the commissionee has joined the operational network, the commissioner performs *operational discovery* using DNS-Based Service Discovery (DNS-SD) to learn the operational IP address of the commissionee. All following communication is performed after a CASE (see Section 3.3.2) was performed, which is used to establish secure connections between nodes in the fabric. At this stage, the commissioner can request the commissionee for *disarming fail-safe*, by asking it to remove the stored configuration backup [65]. The last stage is the *message exchange*, during which commissioner and commissionee exchange an AES-encrypted message on the operational network [65]. Figures 3.3 and 3.2 summarize the commissioning process.

After the commissioning process, the commissionee has obtained the following items: (1) an instance name generated by the commissioner using the fabric ID and (operational) node ID of the commissionee, (2) a NOC valid in the fabric for which the device was commissioned, (3) a private key for the NOC, (4) a new Access Control List (ACL) and (5) information about the operational network (e.g., for Wi-Fi: SSID and passkey).

Password Authenticated Session Establishment

The Password Authenticated Session Establishment (PASE) secures the initial key exchange during the commissioning process using the SPAKE2+ algorithm (see Section 2.3.2), which is a reinforced version of the PAKE protocol run between two parties for deriving a strong shared key over a non-secure channel without risk of disclosure [65] [82]. A variant of the algorithm has been proven to be secure in a preprint by Shoup in 2020 [79], whose work was supported by Apple, one of the CSA's members. PASE messages are sent over reliable channels [30, Chapters 4.13.1.1, 4.13.1.2].

During the PASE, the commissionee's setup code and PBKDF parameters are exchanged to derive 128-bit unidirectional shared secrets `I2RKey` and `R2IKey` from the 27-bit setup code using a PBKDF (PBKDF2). These secrets are used to secure the subsequent communication between the commissioner and the joining device (commissionee) [47] [30, Chapters 2.8, 3.6, 4.12.1.3]. Because SPAKE2+ is asymmetric, only the commissioner is required to know the passcode, while the commissionee has stored the precomputed values for the PAKE response. The PAKE must be completed before any encrypted application data is sent. During the PASE, the commissioner generates a random number and a node-unique Session ID, and then chooses a passcode identifier (*0* for Matter v1.0). The commissionee responds with the corresponding PBKDF parameters (salt and iterations) unless they are already known to the commissioner. The remainder of the SPAKE2+ exchange is described in Section 2.3.2. If the verification fails, the session establishment is aborted and an error report is returned. On success, a `PakeFinished` message is sent which contains a confirmation tag. After verification of the PAKE exchange, both parties can derive unidirectional keys `I2RKey` and `R2IKey` and an attestation challenge that is used for the Device Attestation (DA) Procedure from the SPAKE2+-derived shared secret [30, Chapter 4.13.1.2].

Bluetooth Low Energy

Matter uses Bluetooth Transport Protocol (BTP) to define Matter-over-Bluetooth Low Energy (MATTERoBLE), as demonstrated in Figure 22 of the Matter specification [30, Figure 22]. Each Matter message over BTP forms a Service Data Unit (SDU), that can consist of multiple BTP segments [30, Chapter 4.17]. Bluetooth Low Energy (BLE) focuses on low power transmissions with low latency which makes it ideal for transferring small payloads. The same 2.4 GHz radio frequencies as for Thread and Wi-Fi are used, which allows to use the same antenna. BLE messages are secured using 128-bit AES end-to-end encryption in Counter with CBC-MAC (CCM) and show as small, potentially unidirectional bursts that use frequency hopping spread spectrum.

3.3.2 CERTIFICATE AUTHENTICATED SESSION ESTABLISHMENT

Certificate Authenticated Session Establishment (CASE) is used to secure the exchange of NOCs [65] and salts [48] between nodes in a fabric to negotiate new keys for symmetric encryption [65] and for mutual authentication [48]. During the CASE, a SIGn-and-MAc (SIGMA) algorithm is used [65], a family of algorithms which was introduced by Krawczyk in 2003 [50]. For the CASE, both nodes have to own NOCs chaining back to the same root of trust. After the secure channel has been established and the keys exchanged, AES is used for shared key cryptographic operations [65] using the CASE-derived keys (see Section 2.3.1). Whenever a device breaks the connection, the CASE is reset. Furthermore, a resumption mode allows faster continuation of previous sessions [30, Chapters 4.13.2.2].

3.3.3 COMMUNICATION WITHIN A FABRIC

Matter differentiates between five device types (which are not mutually exclusive) [65]:

Node Devices that have been commissioned to a fabric are nodes. More precisely, one physical device can implement one or more nodes. Each node has a fabric-unique 64-bit Node ID, a 16-bit Vendor Identifier (VID) uniquely identifying a product manufacturer [30, Chapter 2.5.2], a 16-bit Product ID (PID) uniquely identifying a product of a vendor [30, Chapter 2.5.3] and a 16-bit Group Identifier (GID) identifying a set of nodes in a fabric [30, Chapter 2.5.4]. Nodes are identified using a three-element tuple `(Root PK, Fabric ID, Node ID)`.

Controller Controllers are nodes that can act as commissioners during the commissioning process.

Bridge Bridges are network devices that securely connect Non-Matter devices to a Matter fabric (see Section 3.3.8).

Edge Router Edge routers, sometimes also referred to as border routers or access points, are network devices that make different IPv6 networks interoperable, for example by enabling communication between Thread- and Wi-Fi devices (see Figure 3.4).

OTA Provider OTA providers are nodes that can respond to queries from OTA requestors about available software update images and provide them with the update packages in the Over-the-air (OTA) software update process.

OTA Requestor OTA requestors are nodes that are able to request and receive OTA software updates, and handle announcements about available OTA providers in the Matter network.

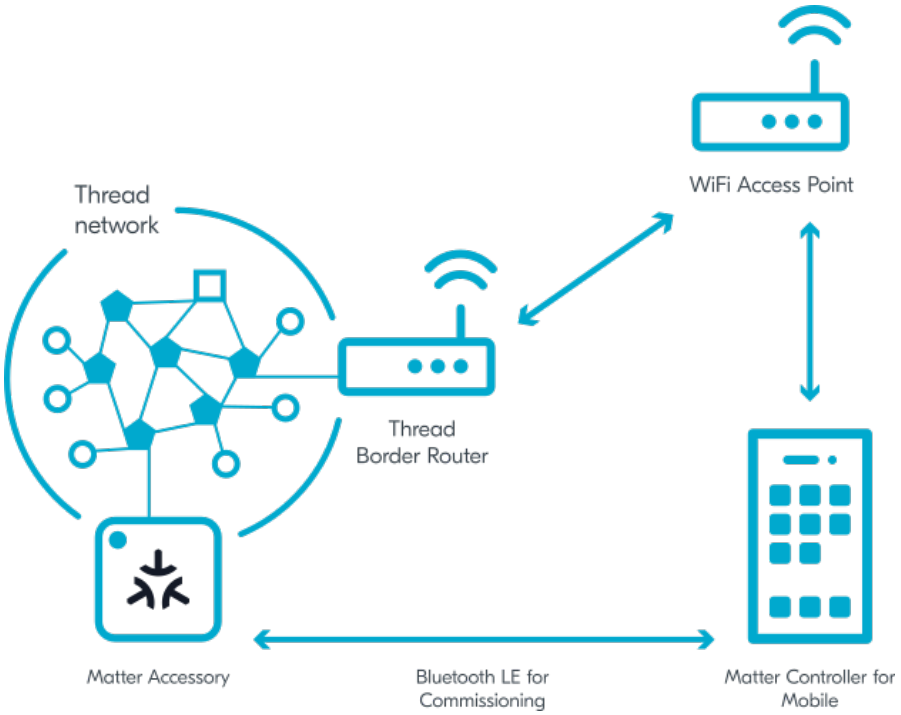


Figure 3.4: Example of the connectivity required between network components to allow communication between Matter devices. The image is unmodified and licensed under Apache License 2.0. Source: Matter code repository [27]

Nodes in a Matter fabric (network) can directly talk to each other over IPv6 [30, Chapter 2.3.2]. All unicast node-to-node messages are secured and authenticated, and provide replay authentication [30, Chapter 2.7]. Additional IPv4 support is optional [30, Chapter 2.5.6.4]. Messages are either control or data messages, with control messages reserved for internal protocols such as the Message Counter Synchronization Protocol (MCSP) for the purpose of security initialization. They are identical in format, but use separate message counter domains, and can therefore operate securely over the same security key [30, Chapter 4.1.1.1]. Whenever a new device has joined the Matter fabric, the commissioner node can inform other nodes about the addition of a new device and instruct them to command it, as well. Matter messages are sent over Message Reliability Protocol (MRP), which is reliable UDP, or TCP for operational messaging, or BTP for commissioning [30, Chapter 4.1.1.2], though commissioning can occur over Wi-Fi (UDP, TCP) as well, if the device is already in the fabric.

Device interaction can be accomplished by *synchronous control* or *asynchronous notification*. For synchronous control, the client sends a command to the server, which performs the action and responds with a status response. Asynchronous notification, on the other hand, uses a publish-subscribe pattern [33] where a client subscribes to event data or cluster attributes of the bound server [30, Chapter 8.2.2]. Nodes can validate the identity in form of NOCs of other nodes within an ecosystem using the Root CA Certificate (RCAC) associated with the ecosystem, and use the ACL to check whether the device is allowed to perform the specific action [81]. Each device has stored an ACL which controls which nodes are allowed to communicate with the device and which operations they may perform, for example limit which devices are allowed to change its configuration or perform the unlock door event [47]. Devices sharing the same root certificate trust each other and form an ecosystem. Once the certificate information exchange and validation of trust are successful, the devices establish a CASE-encrypted session [48].

Devices can form groups, for example a security system, and groupcast (also referred to as multicast) is used to communicate with multiple devices. The site-local multicast address of a group can be derived from the fabric ID and group ID [30, Chapter 2.5.6.2]. The commissioner configures one or more shared keys for each group in the fabric [30, Chapter 2.5.4.1]. Devices within a fabric may belong to different networks and share a root of trust, the RCA issuing the NOCs, also referred to as operational root of trust[30, Chapter 2.4], a configuration state and a unique 64-bit fabric ID. Each node is uniquely identified in a fabric by an 64-bit (operational) node ID [30, Chapter 2.5.5.1], and uses one node ID for each fabric [65]. The unspecified node ID `0x0000_0000_0000_0000` is reserved and should not appear in any messages or protocol usages, since it marks the existence of uninitialized, missing, or invalid node IDs [30, Chapter 2.5.5.6]. Communication across network boundaries always passes Border Routers [30, Chapter 2.3.2].

3.3.4 DEVICE DISCOVERY

Nodes which are already part of a fabric shall use DNS-SD over IP for announcements, if the device is already connected to an IP network it shall perform discovery over IP. Otherwise, device discovery is performed over BLE or Wi-Fi Soft-AP technologies. For this, standard Wi-Fi AP advertisement and connection protocols are employed [30, Chapter 5.4.1]. Wi-Fi Soft-AP is used with a fixed IP address and a SSID which is derived from the device discriminator, the VID and PID, and a per-boot random Basic Service Set Identifier (BSSID). PID and VID may be set to zero for privacy purposes [30, Chapters 5.4.2.6.2, 5.4.2.6.5].

A device has to use multicast Domain Name System (DNS) on Wi-Fi and Ethernet networks to make itself discoverable. On Thread networks, devices use Service Registration Protocol (SRP) and an Advertising Proxy (AdProx) running on a Thread Border Router to make themselves discoverable [30, Chapter 5.4.2.7]. Devices that primarily and mainly control a fabric are required to start announcing when power is applied, other than devices that do not mostly control the fabric and devices whose primary function is some sort of access barrier or lock, which shall not start announcing when power is applied to avoid network pollution and as a mitigation of privacy threats [30, Chapter 5.4.2.2]. Matter supports service advertising and discovery in form of DNS-SD to advertise the IP addresses and supplementary information, such as identifiers, device types and pairing hints of a service. Thread mesh networks use unicast DNS for DNS-SD since Thread is specifically

designed for low-power IoT devices, where the excessive use of the default zero-configuration multicast DNS would consume too much computational power and energy. Thread Border Routers advertise available SRP servers and implement an DNS-SD discovery proxy to allow Nodes within the Thread mesh to discover multicast-advertised nodes [30, Chapter 4.3]. A Matter device may advertise commissionable node discovery service records permanently, although the recommended default behavior is to limit the service announcements. Devices are required to allow the user to configure a timeout [30, Chapter 4.3.1.2]. Commissioning apps may apply arbitrary filters on commissionee devices [30, Chapter 4.3.1.3]. The DNS-SD `TXT` record for the Distinguished Name (DN) is editable by the user within a limit of 32 bytes and Unicode Transformation Format 8 (UTF-8) encoding. Manufacturers are free to include the field during commissioning, but it is recommended not to use the field unless required [30, Chapter 4.3.1.9]. If a device wishes to announce an anonymized PID, it can use 0x0000 which is reserved for this purpose, though it may also be used to indicate that an update applies to multiple PIDs [30, Chapter 2.5.3].

The discovery information consists of the discriminator, also referred to as *friendly name*, and optionally VID, PID and extended data. The extended data (if present) contain a rotating device identifier which manufacturers can use for costumer support purposes, for example to identify the factory software version and related features, manufacturing date or setup code recovery. The feature is optional, but if implemented, the rotating device identifier must be irreversible, must not contain a fixed identifier, is unique per-device, 64 to 256 bits long and considered non-trackable, because it is rotated to protect the user's privacy [30, Chapter 5.4.2.4.5]. It can be used by the customer support to help users with pairing or establishing Node provenance and can be used to programmatically obtain the Node's passcode, but requires proof of possession before doing so, a procedure which is explicitly out-of-scope of the standard.

3.3.5 MATTER OVER THREAD

Apart from its operation over Wi-Fi and Ethernet, Matter can be operated on a Thread network [30, Chapter 2.3.2]. Thread is a secure wireless mesh network stack for IoT developed since 2014 by Thread Group. It is an open IPv6-based protocol for device-to-device and device-to-cloud connections allowing seamless integration of end devices with the internet, designed for low power operation and is application-layer agnostic. The protocol is built on proven and widely available and supported technologies such as IPv6 and 6LoWPAN at the network layer and runs on the IEEE 802.15.4 wireless protocol, providing reliable message transmission at the link layer. UDP is used at the transport layer. Before connecting to a network, each device must authenticate itself. Thread communication is encrypted and authenticated using 802.15.4 security mechanisms and advertises great scalability. Since 2016, Google and NordicSemiconductors develop an open source implementation of the Thread networking protocol called OpenThread [54].

Thread Networks are scalable multi-hop mesh networks in which each device operates as either a router or end device. Routers are selected dynamically in a Thread network from all routing-capable devices when additional range is needed, not requiring any user interaction. In comparison to native Thread devices, Matter-over-Thread devices have an increased memory footprint due to the additional BLE implementation used for commissioning and the mandatory support for OTA

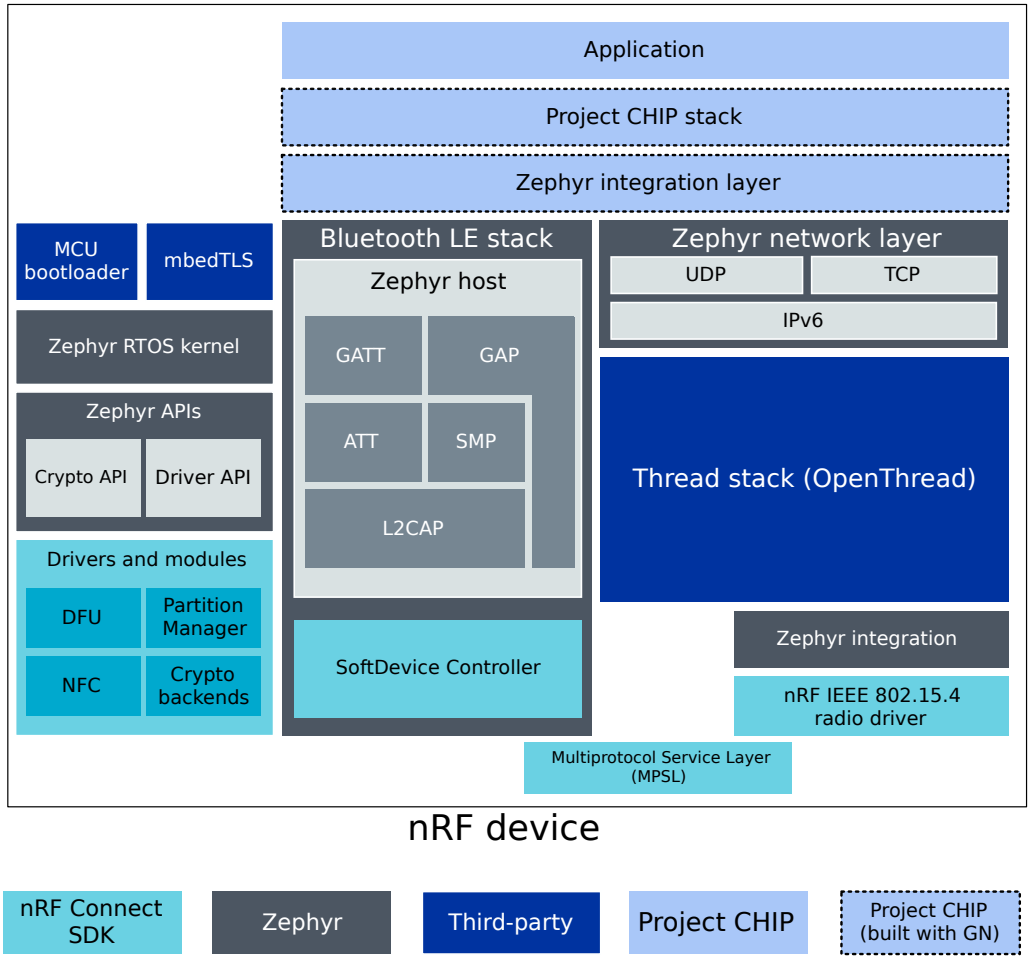


Figure 3.5: Simplified structure of a Matter application running on the nRF Connect platform and using BLE and Thread stacks for communication purposes. The image is unmodified and licensed under Apache License 2.0. Source: Matter code repository [27]

DFU. Figure 3.5 shows a simplified structure of a Matter application running on the nRF Connect platform and using BLE and Thread stacks for communication purposes:

Thread communication is encrypted using AES-128-CCM with 128-bit AES-CBC [56] and uses a Diffie-Hellman (DH) key exchange with Elliptic Curve Cryptography (ECC), using a network-wide key, which should be protected with Transport Layer Security (TLS) or Datagram Transport Layer Security (DTLS) on application layer. In Matter, the network-wide key is distributed in the PASE-protected session.

3.3.6 SLEEPY END DEVICE

Matter nodes can be run in the Sleepy End Device (SED) operating mode to extend and optimize battery life. SEDs can be in either *Idle* (slow-polling) or *Active* (fast-polling) mode, the latter of which the device switches to when actively engaging in an exchange[30, Chapter 2.9].

3.3.7 MESSAGE FORMAT

A 4-bit field defines the Matter message format version [30, Chapter 4.4.1.2], implicating that the standard foresees up to 16 revisions of the message format, after that, other means have to be found to signal the version. An 8-bit field is reserved for the security flags, consisting of a bit for privacy (P), control message (C) and message extension (MX), followed by 4 reserved bits and 2 bits or the session type [30, Chapter 4.4.1.4]. Similar to the IP message format, the Matter message format defines fields for the source and destination ID [30, Chapters 4.4.1.6, 4.4.1.7]. A full description of the message format is found in Chapter 4.4 of the Matter specification [30, Chapter 4.4]. Matter nodes are not required to support IPv6 fragmentation and are expected to support 1,280 bytes as Maximum Transmission Unit (MTU), which is the minimum for IPv6. It must be ensured that all messages fit in the MTU, if sent over UDP. When other transport protocols (TCP, BTP) are used, larger message sizes can be used if both nodes support them [30, Chapter 4.4.4].

3.3.8 BRIDGING OF NON-MATTER IOT DEVICES

Using border routers as *bridges*, Matter devices can communicate with Non-Matter devices. Because this feature allows interoperability between devices which were not compatible in the past, it has a great impact on overall device interoperability, but risks a negative impact on device and network security. Typically suitable devices for this task are Wi-Fi access points or home hubs (e.g., smart speakers). Bridges are not limited to their bridging functionality, but may also behave as 'regular' Matter nodes.

Each bridge must implement the device `Bridge` at endpoint 0 and implement the `PartsList` field which lists all endpoints for bridged devices, one endpoint each per bridged non-Matter device [17]. For each endpoint that is listed in the `PartsList` field, the bridge device implements exactly one endpoint that represents exactly one Non-Matter device. Therefore, bridges take the role of translators and expose the bridged devices to the Matter network [65].

3.4 PROTOCOL STACK

The protocol stack of Matter consists of seven layers running on top of TCP or UDP [27], see Figure 3.6. Describing from top to bottom, the *Application* layer defines the high-order business logic of a device. Communication and commands are sent over this layer. The next layer is the *Data Model* which abstracts how application data is structured (see Section 3.4.1).

Whenever a device interacts on the application layer it will operate on these data structures. The *Interaction Model* layer defines the possible interactions between nodes (see Section 3.4.2). Next, the *Action Framing* layer frames the above layers into a prescriptive packed binary format, which is encrypted and signed in the *Security* layer, using proven cryptographic algorithms such as ECC NIST P-256 [65] and AES-CCM-128 [27] and appends a MAC [61]. Hence, this layer adds authentication, confidentiality and protection of all layers above to the protocol stack. Once the packet has been secured, the *Message Framing and Routing* layer creates the payload format and header fields used for routing and specifications of message properties. Lastly, the *Transport Management layer* ensures the message gets delivered by using either TCP or the combination of UDP and Matter's MRP, respectively BLE during device commissioning [61]. In the following, we briefly describe two of the layers, as they are essential to understand how Matter devices operate.

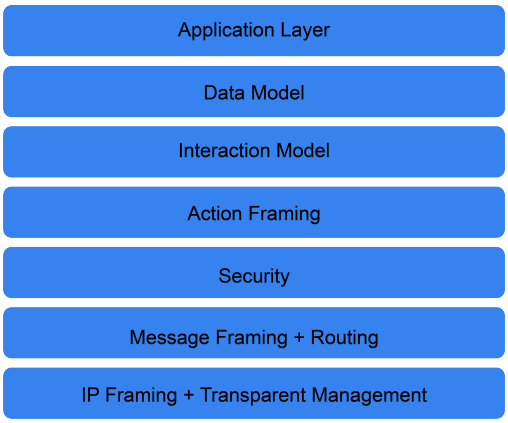


Figure 3.6: The protocol stack of Matter. The image is unmodified and licensed under Apache License 2.0. Source: Matter code repository [27]

3.4.1 MATTER DATA MODEL

The data model of Matter is described in the *Data Model Layer* of the Matter protocol stack. *Nodes* are typically physical devices that the user recognizes as a whole. A *device* may implement multiple nodes, each of which is a complete Matter implementation [61]. Each node can have multiple *endpoints*, which are logically grouped services, also referred to as feature sets [80] [61]. Endpoint 0 is mandatory and reserved for the *Utility Clusters* [61], which are services that apply to the entire node and contain mandatory clusters such as the *Basic Information Cluster Server* that provides basic information about the node such as firmware version or manufacturer, the *ACL Cluster Server*, that allows configuration of the node's ACL and the *Network Commissioning Cluster Server*, that allows network configuration [80]. Each endpoint has one or more *device types*, which indicate a set of commonly available functionality [80]. Furthermore, each endpoint has one or more *clusters*, which group together commonly used functionality in reusable building blocks, such as attributes, commands and events, which in combination provide a single feature [80] [61]. Clusters are either *cluster servers*, which are the devices providing a service, for example holding attributes, commands and events, or *cluster clients*, which are controlling devices that perform interactions with cluster servers [80] [61]. *Attributes* form the device state [61]. *Commands* allow to invoke specific behaviors on a cluster and may have parameters associated with it [80]. *Events* are a special type of attributes that communicate changes of device state [61]. If one or more end points provide a certain set of clusters, the device becomes a certain *device type* [61], some of which are globally defined for Matter.

3.4.2 INTERACTION MODEL

The interaction model of Matter is described in the *Interaction Model Layer* of the Matter protocol stack and defines how to exchange the data defined in the data model between a client and server device. Clients are typically *initiators*, while servers are usually the *targets*. The Interaction Model defines four interaction types [61]:

Read Get the value of attributes or events

Write Modify attribute values

Invoke Send commands

Subscribe Subscribe to periodical data reports (instead of polling the data) regarding all or specific attributes and events

Multiple *actions* form a *transaction*, and multiple *transactions* form an *interaction*.

3.5 CERTIFICATES

The security of Matter communication greatly depends on the certificates used by Matter devices. Therefore, the following two subsections will provide a brief overview of the blockchain used to publicly provide global device, vendor, certification and certificate information, and the two PKIs that are used in Matter.

3.5.1 DISTRIBUTED COMPLIANCE LEDGER

The Distributed Compliance Ledger (DCL) is a distributed blockchain multi-node network maintained by the Test and Certification Oversight Committee (TCOC), which is part of the manufacturing, certification and commissioning process [2] [63] [24].

Trusted Product Attestation Authority (PAA) certificates are stored in the DCL [48]. It provides data integrity and immutability by publishing specific information attesting the provenance and performance of a device. Transactions are individually signed using pre-approved keys. There is no single company in charge of the ledger, and therefore, the data is distributed across different geographical locations [24], hosted on multiple servers each holding a copy of the data by CSA members, communicating with each other using a cryptographically secure protocol [18] and depending on a consensus algorithm, which depends on majority approval.

The DCL holds five types of information: (1) the *Vendor Schema*, providing vendor-specific information, (2) the *DeviceModel Schema*, providing general device information shared across all software versions of a specific product, (3) the *DeviceSoftwareVersionModel Schema*, providing software version-specific information, (4) the *Compliance Test Result Schema*, providing compliance and test result data about the device, and (5) the *PAA Schema*, providing a list of PAA Certificates for the approved PAAs [18].

2 A web UI for the DCL can be found here: https://webui.dcl.csa-iot.org/ [Last accessed: 2022-08-11]

Matter devices can, beside others, use the DCL to check device certification compliance status, verify DACs, getting commissioning instructions, links to manuals, and product information, checking OTA DFU status and upgrade the device to the latest firmware [18]. For example, devices can have a signed firmware that denies all requests until it has been associated with the DCL, from which it inherits the security properties of a specific device class maintained by the manufacturer. Additionally, the DCL can be used to revoke suspicious devices [24].

Read Access to the DCL

The DCL has public read access. Therefore, anyone can read out device model information, including firmware and hardware versions, the device compliance status and PAA certificates [18].

Write Access to the DCL

Write access to the DCL is restricted to CSA and elected entities, as is documented in Section 4.1.4 and following sections. While the CSA can write or revoke the compliance status of a device model, only new device models belonging to the VID of the vendor account trying to add the device model can be added to the DCL, and only vendors can update a subset of information about their own device models. During vendor account creation process, the Vendor Identifier is associated to the vendor public key [18].

3.5.2 PUBLIC KEY INFRASTRUCTURE

The PKI of Matter can be divided into two parts: (1) The Device Attestation (DA) PKI rooted in CSA, which is globally in effect for all Matter devices, and (2) the operational PKI within a specific home network.

The global PKI is overseen by the TCOC, who appoint the Product Attestation Authority (PAA), a set of selected organizations that form the root of trust for Matter devices and act as a CA. The certificates of the PAA are self-signed. The Product Attestation Intermediate (PAI) is responsible for issuing the Device Attestation Certificate (DAC) and owns a certificate signed by the PAA. PAA and PAI each use a key pair consisting of a public and a private key for signing the certificates. The PAA may sign any DAC, but the PAI may only sign DACs of a single vendor. One vendor may operate multiple PAIs. In this way, each certificate chain of a DAC has a root of trust in a PAA, whose certificates are stored in the DCL, which is also provided by CSA's TCOC. Matter devices are shipped with

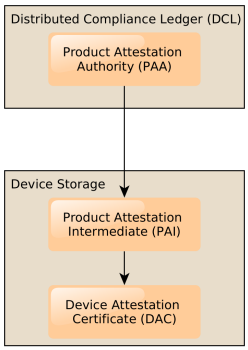


Figure 3.7: The Product Attestation Authority is stored in the Distributed Compliance Ledger, while the Product Attestation Intermediate and Device Attestation Certificate are stored in the device storage [48]

write-protected *factory data partitions*, separate from the firmware and unique to each device instance, containing the PAI certificate and the DAC, the latter of which is unique to each device. The PAA is stored in the DCL, while PAI and DAC are stored in the device storage, see Figure 3.7 for a visualization. CSA members are provided with a Vendor Identifier (VID), which is a unique number identifying the manufacturer. To obtain a Certification Declaration (CD) for a specific device type, members can undergo the certification process. Once the certification is passed, the CSA member can choose to either request signing of the PAI and DAC by one of the PKI providers of the CSA members, or they can set up their own PKI, by which they become their own PAA. Another option is to develop the device with Systems-on-Chips (SoCs) that have baked-in DACs. In this case, the VID and Product ID (PID) in the DAC do not necessarily match those of the device manufacturer [63]. All private keys must be protected, otherwise all signed certificates of the respective certificate will lose their value. The operational PKI used in the home network roots in the operational Root CA Certificate (RCAC), which is commonly owned by a Matter controller, acting as a commissioner. The commissioner, who owns either a RCAC or ICAC, issues and signs the Node Operational Certificate (NOC), which uniquely identifies a node in a Matter fabric [81]. This way, nodes in a fabric can recognize and trust other devices in a fabric. The operational root of trust uses a globally unique public key [30, Chapter 2.4]. Certificates use a X.509 certificate encoding.

3.6 SECURITY FEATURES

Matter claims to be secure-by-design and values security as an important feature to prevent unauthorized device access in general and their integration into botnets, such as Mirai, which can result in devastating Distributed Denial-of-Service (DDoS) attacks [54] [23]. Explicit risks mentioned by CSA are product malfunctioning through remote control, DDoS, data and privacy breaches, theft of intellectual property and potential harm to humans, malicious code injection during manufacturing, physical and remote attacks during operation, and unsecured device maintenance [29].

According to the CSA [29], Matter embeds the following five properties into its security principles: comprehensive, strong, easy-to-use, resilient and agile. To achieve *comprehensiveness*, a layered approach is used with authentication and attestation for commissioning to protect every message and secure a Over-the-air Device Firmware Update (OTA DFU). The CSA claims that Matter's security is self-contained and does not rely on the security of the communication technologies on top of which Matter runs [29], naming integrity and confidentiality as examples [3]. Furthermore, the CSA claims that no functional security features need to be added on top of what is defined by the Matter core specification. To achieve *strength*, Matter uses strong and well-tested standard cryptographic algorithms with strong keys and True Random Number Generators (TRNGs), PASE and CASE for onboarding, attestation and operation, DA so that Matter devices cannot join a fabric unless proven genuine, as well as the DCL used to check whether a device is Matter certified. *Ease of use* is addressed by making it easy for developers to implement secure devices, by providing examples and test vectors for each functional security aspect and reference implementations

3 This is only partially true, as obviously, the availability of Matter must be dependent of the availability of any communication technologies used. If the Wi-Fi used for the operational Matter fabric is DoSed, all Matter communication is DoSed. We assume a linguistic impreciseness and will focus on integrity, confidentiality and authenticity in subsequent considerations of underlying communication technologies.

using the functional security of Matter, including examples for the integration of Hardware Security Moduls (HSM). Furthermore, the user should not be required to think about security at all. For *resilience*, Matter allows to resume secure sessions using a shorter secure resumption protocol instead of running the full protocol each time a session is reestablished. Furthermore, DoS prevention has been integrated, resilience should also apply to SEDs and group communication, for example by using sophisticated message counters. Additionally, firmware integrity attestation is recommended, providing measured boot and attested firmware measurements. *Agility* is achieved by having crypto-flexibility in mind. All cryptographic primitives are abstracted, so that security fixes of the modular designed core specification can be integrated without requiring to change the entire specification.

Matter contributes to upholding the following privacy principles for interaction within their scope: lawfulness, fairness and transparency, purpose limitation, data minimization, accuracy, storage limitation, cybersecurity (integrity and confidentiality) and accountability . It does that by incorporating the following security measures: All data communication between Matter devices have the highest level of confidentiality and integrity supported by current civilian standards of network communications to prevent eavesdropping and tampering. Furthermore, all Matter devices provide proof of identity, so that data is only shared between known Matter entities, Matter is publicly accessible as an open standard, allowing peer-review and validation, data-minimization for Matter interactions and data is bound to specific contexts.

Furthermore, the following privacy preserving mechanisms were incorporated: unique random node identifiers, session establishment with configurable privacy, non-trackable IP addresses and private message headers for sessions, specifically separate privacy encryption keys protecting the identity of the communicating parties [29].

Platform security is addressed, by giving manufacturers room to select the appropriate platform serving their individual needs, as long as it does not endanger the ease of use and functionality of Matter devices. For example, door locks in public area should protect against remote network attacks and be tamper resistant, but light bulbs are not required to be tamper resistant. Manufacturers are free to implement the platform themselves, or reuse a Matter-ready hardware platform.

Additionally, the following security promises were made in an official CSA video [7]: Mutual authentication [4], multiple administrators and controllers and verified access controls to prevent unauthorized actions. In addition to the before-mentioned security mechanisms, Infineon states that Matter follows the following security tenants [46]: No common or default passwords, keep software up to date and ensure software updates are authenticated and validated and monitor, identify and fix vulnerabilities as they are found. We summarized Matter's security claims and our evaluation in Appendix A.

More specifically, every Matter device has the following items installed during manufacturing process [7] [47] [46] [63] [48]: a DAC, including a VID and PID, a private key for the DAC and the PAI certificate used to sign the DAC, a CD and a verifier and onboarding payload unique to the device used for device commissioning. During commissioning, the following items are installed: the fabric-specific NOC including the corresponding RCA or ICA, network credentials and the initial ACL configuration.

4 Matter Security and Privacy: A Deep Dive with the Experts - Connectivity Standards Alliance Timestamp 33:10. The other claims are visually perceivable [7].

To counter security threats, the Matter specification includes countermeasures against threats, which are further discussed in Section 4.2.25. The CSA's threat model regards the following attributes of an attacker [7]:

- Motivation (political, personal, financial, indirect)
- Capabilities

  Resources (Time, money, tools, ...)

  Expertise (Skills, knowledge, ...)

  Access (Physical, proximity, remote, ...)
- Role (user, former user, guest, supplier, trusted party, intruder, ...)
- Lifecycle (former owner, new owner, scavenger, ...)

The CSA's threat analysis respects the following items [7]:

- Severity based on Likelihood and Impact
- Likelihood - Probability based on

  Access (Physical, Proximity, Remote) and

  Difficulty (Easy, Moderate, Difficult)
- Impact - Effect of successful attack based on

  Scope (Single Device, Home Network, Fleet) and

  Data and Control (Low Sensitivity, Limited Sensitivity, Complete Compromise)

4 SECURITY ANALYSIS

Most IoT devices that lack security by design simply pass the security responsibility to the consumer, thus, treating the customers as techno-crash test dummies.

— JAMES SCOTT

4.1 SECURITY REQUIREMENT ENGINEERING

Engineering the security of a system is about the identification of security risks, security requirements and recovery strategies. Incorporating security into the system as early as possible can save time and costs as well as reputation when compared to retro-fitting security into a system that was not designed for security from the start due to architectural constraints. In 2005, Myagmar et al. [57] introduced threat modeling as a basis for security requirements and proposed to approach the security requirement engineering in a three step process (the phases may interleave and iterate):

1. Threat Modeling
2. Development of Security Requirements
3. Development of Security Mechanisms

They state that threat modeling can be used to justify security countermeasures and validate the assumptions made by system architects. For the security requirements engineering part of this thesis, the following process will be conducted, as described by Myagmar [57]:

4.1.1 THREAT MODELING

In the first step of threat modeling, the protocol is characterized using a Data Flow Diagram (DFD). Second, the assets and access points are identified [57]. The OWASP Threat Modeling Process (TMP) provides examples and further differentiates between entry and exit points [22]. For this, potential adversaries, their resources, access, risk tolerance, objectives, their motivation and goals and the amount of their inside information must be determined. In addition, trust boundaries and trust levels have to be identified [57]. In the third step threats are identified. A threat is the adversary's goal and can be identified by going through each of these security critical entities and creating threat hypotheses that violate confidentiality, integrity or availability of the entity. To start threat modeling, lists of known threats and vulnerabilities found in similar protocols can be used to step through each of the assets and check whether they apply. The result of the threat identification process is a *threat profile* for the protocol [57]. Each of the threats can be characterized in different threat classes. A program commonly used for threat modeling, OWASP Threat Dragon, offers three different characterizations [40]:

STRIDE Spoofing, Tampering, Repudiation, Information disclosure, Denial-of-Service, Elevation of Privilege

CIA Confidentiality, Integrity, Availability

LINDDUN Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness, Non-Compliance [97]

To determine whether a system is susceptible to a threat, attack graphs can be used [57]. The TMP recommends listing external dependencies and using threat trees and abuse case graphs [22].

4.1.2 DEVELOPMENT OF SECURITY REQUIREMENTS

In general, requirements specify what a system does, but not how. Security requirements specify what must not happen, where the event that must not happen is usually a threat. Since security and usability are a trade-off, it is important to discover which threats are realistic and must be dealt with. For this, metrics are used. Myagmar et al. [57] propose a metric where the number of affected users, damage potential, level of skills needed, cost of attack, reproducibility and discoverability are rated on a scale from 1 to 10, and the average is calculated as the total risk for each threat. The metric is similar to the DREAD model, which uses damage potential, reproducibility, exploitability, affected users and discoverability instead. Since ratings like these are subjective, the TMP suggests using a qualitative risk model instead, where qualitative values such as high, medium and low are assigned to various aspects [22].

The next task is to specify security requirements. Note that *security policies* are sets of security requirements. Common security requirements can be adapted from standards such as Common Criteria (CC) [21] [57]. To check the completeness of the protocol's security, threats and requirements (rsp. countermeasures) can be mapped to each other. In addition, the mapping can be used to justify security mechanisms. The requirements engineering process can be divided into three steps [57]:

1. Requirements elicitation (convert threats into "shall not happen" requirements)
2. Requirements analysis and negotiation
3. Requirements validation

Risk management addresses the the circumstance, that it is impossible to provide 100% security and balances what is possible and what is acceptable in terms of countermeasures. The idea is, that instead, 100% risk acceptance can be reached. Myagmar et al. advocate, that designer should ensure that security does not become more expensive than it is worth [57]. While it can certainly be agreed on, that safety is a very important feature, it is worth of discussion how important the user's privacy is and how to weigh up security, safety and privacy for protocols that can be used in arbitrary contexts with (almost) arbitrary devices and networks from arbitrary individuals of unknown worthiness of protection. Their proposed risk management consists of [57]:

Risk assessment prioritize threats by the factors damage and likelihood

Risk reduction either transfer the risk to another entity (e.g. to the user), remove the risk (e.g. remove component associated with it) or mitigate the risk by deploying countermeasures

Risk acceptance accept the risk

4.1.3 DEVELOPMENT OF SECURITY MECHANISMS

The development of security mechanisms is done by proposing mitigation measures for each threat, for example by describing firewall settings or resource access restrictions [57].

4.1.4 THREAT MODEL

The following subsections summarize our threat model of Matter.

External Dependencies

External dependencies of Matter are Wi-Fi, Ethernet, Thread, DNS-SD, BLE and BTP, Dynamic Host Configuration Protocol (DHCP), Secure Hypertext Transfer Protocol (HTTPS), X.509v3 certificates, Network Time Protocol (NTP) and the specified cryptographic algorithms, which are discussed in Section 4.2.1.

Table 4.1: External Dependencies

Dependency	Risk
Wi-Fi, Ethernet, Thread	DoS of operational communication, eavesdropping on (encrypted) communication and CASE
BLE, BTP	DoS of commissioning, eavesdropping on PASE
DNS-SD, DHCP	Availability of operational communication, pairing with impersonated device
HTTP	DoS of device attestation validation, optionally DoS of commissioning and OTA DFU
X.509v3 certificates	Device impersonation on the operational network, counterfeits in the DCL
NTP	Control validity of certificate and encryption key
Cryptographic primitives	Information Disclosure, tampering, privacy risks, impersonation, have the potential to fully compromise Matter's security

Protective Items

We determined lists of resources that are typical for Matter systems, to check their worth for the attacker (see Appendix B). Subsequently, we defined in how far these resources require protection:

Table 4.2: List of what systems require protection

ID	Name	C	I	A	STRIDE
P.1	DCL		X	X	STDE
P.2	Certificates and Public Keys		X		ST
P.3	Private Keys	X	X		TI
P.4	Controller		X	X	STIDE
P.5	Edge Router		X	X	STIDE
P.6	Node		X	X	STIDE
P.7	Device service			X	STD
P.8	Device firmware	(X)	X		TIDE
P.9	Device ACL	X	X		TE
P.10	Communication over Matter	X	X		STRID
P.11	Matter- and Non-Matter network			X	SDE
P.12	OTA Provider		X	X	STIDE
P.13	OTA Requestor		X	X	STIDE
P.14	Bridge		X	X	STIDE
P.15	Onboarding payload	X	X	X	TI
P.16	Device firmware update	(X)	X	X	STIDE
P.17	Device hardware	(X)	X	X	STRIDE
P.18	Device configuration		X		TE
P.19	Certification Declaration		X	X	STD

Trust Levels

Table 4.3: Trust Levels

ID	Name	Description
TL.1	TCOC	A member of the Test and Certification Oversight Committee who can elect PAA and issue CDs
TL.2	Trustee	An appointed CSA member who approves or disapproves other DCL roles
TL.3	Validator Node (VN) Administrator	An authorized CSA member who hosts a copy of the DCL and participates in the consensus protocol
TL.4	CSA Certification Center (CSA CC)	Submits certification status to the DCL as provided by a Test House
TL.5	Test House	Perform certification tests and provide test results
TL.6	Observer Node (ON) Administrator	A CSA member who runs an ON with read-only access to the DCL
TL.7	Vendor with PAA	A manufacturer owning a Product Attestation Authority who can issue PAI certificates and DACs
TL.8	Vendor with PAI	A manufacturer owning a Product Attestation Intermediate who can issue DACs
TL.9	Vendor without PAA and PAI	A vendor who sells Matter-certified components and device
TL.10	User with access to controller	A user with access to a Matter controller can create fabrics, assign RCA and ICA to nodes, configure ACLs
TL.11	User with access to a controller, a node and its onboarding payload	A user with access to a controller, node and its onboarding payload can add the node to a fabric in their control, and inherits all rights from TL.10
TL.12	User of a bridge	A user who owns a Matter bridge has the rights to connect bridge-compatible Non-Matter devices to the fabric
TL.13	User of an OTA Requestor	A user who owns an OTA Requestor can query the DCL to check for software updates
TL.14	User of an OTA Provider	A user who owns an OTA Provider can query an OTA Requestor to check for software updates, and distribute them to other nodes
TL.15	User of an Edge Router	A user who owns an Edge Router can make devices of two distinct Matter-certified and IPv6-based communication technologies interoperable

Continued on next page

Table 4.3: Trust Levels (Continued)

ID	Name	Description
TL.16	User of a Node	A user who uses a Matter node without administration rights, which is neither a bridge, edge router, OTA requestor or OTA provider
TL.17	User in the underlying network	A user who is not in a fabric, but is connected to the underlying network

Assets

During threat modeling, the following assets were identified:

Table 4.4: Assets (the attacker is interested in)

ID	Name	Description	Trust Level
A1	Home Network	Assets relating to the home network	
A1.1	Ability to issue commands to a device	This is the ability to successfully issue commands to devices in the Matter network	(10) User with access to a controller, (11) User with access to a controller, a node and its onboarding payload, (12) User of a bridge, (16) Anonymous node user
A1.2	Ability to modify device ACL	This is the ability to successfully modify the ACL of another device in a Matter network	(10) User with access to a controller, (11) User with access to a controller, a node and its onboarding payload
A1.3	Ability to add/remove a device to/from a fabric	This is the ability to successfully add/remove a device to/from a fabric by being able to issue/revoke a NOC to/for a device and a given fabric. This ability belongs to Matter controllers during the device commissioning process and would allow an individual to add arbitrary Matter-devices to the Matter network	(11) User with access to a controller, a node and its onboarding payload, (12) User of a bridge

Continued on next page

Table 4.4: Assets (the attacker is interested in) (Continued)

ID	Name	Description	Trust Level
A2	DCL	Assets relating to the DCL and certification	
A2.1	Ability to issue a DAC	This is the ability issue a valid DAC to a specific device	(7) Vendor with PAA, (8) Vendor with PAI
A2.2	Ability to add, remove or modify a device type	This is the ability to successfully add, remove or modify a certified device type by adding or revoking a CD to/from the DCL	(1) TCOC, (3) Validator Node (VN) Administrator
A2.3	Ability to participate in the consensus protocol	This is the ability to successfully participate in the consensus protocol	(3) Validator Node (VN) Administrator
A2.4	Ability to grant or revoke rights to participate in the consensus protocol	This is the ability to successfully grant or revoke rights to participate in the consensus protocol	(2) Trustee
A2.5	Ability to influence DCL content	This is the ability to influence the contents of the DCL, regardless of the quantity and quality	(1) TCOC, (3) Validator Node (VN) Administrator, (4) CSA Certification Center (CSA CC), (5) Test House
A2.6	Ability to add/remove a PAA to/from the DCL	This is the ability to successfully add a PAA to the DCL	(1) TCOC
A2.7	Ability to issue/revoke a PAI	This is the ability to successfully issue or revoke a PAI	(7) Vendor with PAA
A2.8	Availability of the DCL	The DCL should be available at all times and can be accessed by all Validator Node, Observer Node and commissioners	(1) TCOC, (3) Validator Node (VN) Administrator, (6) Observer Node (ON) Administrator

Continued on next page

Table 4.4: Assets (the attacker is interested in) (Continued)

ID	Name	Description	Trust Level
A2.9	Ability to read-access the DCL	This is the ability to retrieve information from the DCL	(3) Validator Node (VN) Administrator, (6) Observer Node (ON) Administrator, (10) User with access to a controller, (11) User with access to a controller, a node and its onboarding payload [6]

Access Points

Table 4.5 lists access points that are both entry and exit points, due to the partially direct device communication over the operational network or by physical proximity:

Table 4.5: Access Points

ID	Name	Description	Trust Levels
AP.1	Pairing Process	All devices in a fabric undergo an initial pairing process, where devices use a BLE-secured session to perform a PASE	(11) User with access to a controller, a node and its onboarding payload
AP.2	Device Discovery	Commissioners and nodes discover other devices by using DNS-SD over Thread, Wi-Fi or Ethernet	(10) User with access to a controller, (11) User with access to a controller, a node and its onboarding payload, (12) User of a bridge, (13) User of an OTA Requestor, (14) User of an OTA Provider, (15) User of an Edge Router, (16) User of a node, (17) User in the underlying network

Continued on next page

Table 4.5: Access Points (Continued)

ID	Name	Description	Trust Levels
AP.3	Operational Communication	Nodes communicate in the operational network using encrypted messages	(10) User with access to a controller, (11) User with access to a controller, a node and its onboarding payload, (12) User of a bridge, (13) User of an OTA Requestor, (14) User of an OTA Provider, (15) User of an Edge Router, (16) User of a node
AP.4	Operational Session Establishment	Nodes that establish a connection perform a CASE within the operational network. Trust level refers to everyone who is potentially able to initiate the CASE	(10) User with access to a controller, (11) User with access to a controller, a node and its onboarding payload, (12) User of a bridge, (13) User of an OTA Requestor, (14) User of an OTA Provider, (15) User of an Edge Router, (16) User of a node, (17) User in the underlying network

Table 4.6 lists entry points:

Table 4.6: Entry Points

ID	Name	Description	Trust Levels
EP.1	Over-the-air Device Firmware Update	Vendors can submit device updates to the DCL	(7) Vendor with PAA, (8) Vendor with PAI
EP.2	DCL write-access	Vendors and CSA publish information about devices in the DCL, some of which directly influence device and network security	(7) Vendor with PAA, (8) Vendor with PAI

Continued on next page

Table 4.6: Entry Points (Continued)

ID	Name	Description	Trust Levels
EP.3	In-field Upgrade to Matter	Non-Matter devices can become Matter-devices via an In-Field Upgrade. For example, the vendor Eve uses iPhones to in-field upgrade some of their Matter devices	(7) Vendor with PAA, (8) Vendor with PAI
EP.4	Device Passcode	The user enters another device's passcode into the commissioner to pair with it	(11) User with access to a controller, a node and its onboarding payload
EP.5	DNS response	The user can modify the Distinguished Name field which is advertised in DNS records	(10) User with access to controller
EP.6	Legacy out-of-band updates	Devices are allowed to perform legacy out-of-band updates	(16) User of a node

Table 4.7 lists exit points:

Table 4.7: Exit Points

ID	Name	Description	Trust Levels
XP.1	Pairing Hint (PH)	When the user instructs the commissioner to commission a device, he can query the PH for help	(11) User with access to a controller, a node and its onboarding payload
XP.2	Over-the-air Device Firmware Update	Nodes regularly check for software updates	(13) User of an OTA Requestor, (14) User of an OTA Provider

Continued on next page

Table 4.7: Exit Points (Continued)

ID	Name	Description	Trust Levels
XP.3	Device Discovery	Commissioners and nodes discover other devices by using DNS-SD over Thread, Wi-Fi or Ethernet	(10) User with access to a controller, (11) User with access to a controller, a node and its onboarding payload, (12) User of a bridge, (13) User of an OTA Requestor, (14) User of an OTA Provider, (15) User of an Edge Router, (16) Anonymous Node user, (17) User in the underlying network
XP.4	Operational Communication	Nodes communicate in the operational network using encrypted messages	(10) User with access to a controller, (11) User with access to a controller, a node and its onboarding payload, (12) User of a bridge, (13) User of an OTA Requestor, (14) User of an OTA Provider, (15) User of an Edge Router, (16) Anonymous Node user
XP.5	DCL read-access	Commissioners verify device attestation by validating the certification chain using DCL replies, OTA requestors check for software updates	Public, (1) - (17)

Potential Adversaries

Matter is not limited to a set of potential adversaries, due to its high ambitions an arbitrary usage context. However, specific threats or attacks might attract a specific subset of attackers. Attacker types can be categorized in various dimensions, in one of which they are commonly categorized in categories such as script kiddie, expert, cyber criminal, cyber terrorist, nation-state actor (e.g., spy) and so forth. Literature research by Panchenko et al. in 2006 showed [66], that while there are a number of attacker models, most of them are either very simplified or abstract. To our best knowledge, there is no established attacker model that the security community has agreed upon. In 2019, Do et al. [34] conducted a survey of prominent adversary models, that gives a good starting point towards the topic. We leave it to future research to define the various attacker types and assign them to specific threats and attacks, as we believe a generic and multi-purpose classification methodology would greatly help defensive security experts to standardize adversary-specific classification and terminology. In general, we expect an attacker-based threat model to distinguish attacker types by at least their resources in terms of time and material, skill, access, risk tolerance and objectives.

For some of the threats, we assume that the attacker is inside the operational network (e.g., Wi-Fi), because of Matter's claim that its security is independent from the communication technologies it is operated on.

Threats

We identified the following generic adversarial goals, and provided examples of more specific goals:

- Generic adversarial goal: Steal intellectual property

  Extract firmware

- Generic adversarial goal: Steal sensitive user data

  Obtain the target's personal data, e.g., name, IDs such as the advertising ID, contacts

  Obtain the target's device usage pattern, e.g., device is turned on daily at 1:37 pm

  Obtain the target's preferences (political, sexual, behavioral, ...), e.g., demisexual furry

  Obtain the device location, e.g., work place and home [8]

  Obtain biometric, genetic or health data, e.g., facial pattern

  Obtain financial data, e.g., balance

  Obtain classified data, e.g., regarding national security

  Obtain business data, e.g., project names

  Obtain application credentials, e.g., *johndoe:janeyhoney*

- Generic adversarial goal: Steal network credentials to gain direct access to the underlying network

  Steal Wi-Fi SSID and passkey, e.g., *SSID: NetBox! 42, passkey: 7#345vv3970411*

  Steal Thread network credentials, e.g.,
  Master Key: 0x00112233445566778899AABBCCDDEEFF,
  Commissioning Key (PSKc): c3f59368 445a1b61 06be420a 706d4cc9,
  Mesh-Local Prefix: fde5:8dba:82e1:1::/64, Extended PAN ID: 0x0001020304050607,
  Network Name: Test Network [9]

- Generic adversarial goal: Control device service

  Perform actions, e.g., instruct node to activate the camera

  Adjust parameters of the device service, e.g., the pace rate of a pace maker

8 This can be especially, but not exclusively, problematic for military employees: https://www.theverge.com/2018/1/28/16942626/strava-fitness-tracker-heat-map-military-base-internet-of-things-geolocation [Last accessed: 2023-01-11]

9 For out-of-band Thread commissioning, only Network Name, Extended PAN ID and Master Key are required. Source: Thread Specification v.1.1.1 Chapters 11.2, 8.4.6.2

- Generic adversarial goal: Use resources of the target

  Integrate node into a botnet to carry out attacks, e.g., integrate into a Mirai botnet to launch large-scale DDoS attacks

  Use node to brute-force a password / for computation, e.g., as a crypto miner

  Use node as an intermediate to extend range to a target, e.g., hijack private device of a military employee in a (potentially secret) and isolated military base

  Use node to store information, e.g., a secret file used to blackmail a person, that should not be found in the attacker's domain but could be heavily distributed among nodes, or malicious payload

  Use node to distribute malware, e.g., infect them with a computer worm

- Generic adversarial goal: Ongoing espionage on the target

  Exfiltrate sensor data, e.g., camera footage or audio recordings

  Obtain application meta data„ e.g., ISSN of a book, author, title and year

  Obtain deltas to the sensitive user data, e.g., sex changed to divers

- Generic adversarial goal: Use device as jump host to another target

  Attack bridged Non-Matter device, e.g., target a bridged door lock via a Matter bridge that is in control by the attacker

  Attack the network on which Matter operates, e.g., congest the network

- Generic adversarial goal: Denial-of-Service

  Sabotage manufacturing process of competitor by triggering a manufacturing defect, e.g., ship Matter-ready SoCs with Trojans specifically designed to activate only for devices with a specific VID

  Attack active health application, e.g., insulin pump to harm the target's safety

  Cause sensor malfunctioning to harm the target's physical safety, e.g., a passive health-monitoring application or a smoke alarm

  DoS a device, e.g., deny service to the legitimate user, for example by irreversibly stopping a pace maker or deactivating a security camera

  For blackmailing purposes

  DoS the entire fabric, e.g. in the course of cyber warfare against a nation state, to cause substantial financial and structural damage, to outcompete a competitor or for blackmailing

Note that we did not list *harm of Matter reputation* because this goal is achieved by a combination of other attack goals. Furthermore, any attack goals listed are required to apply to Matter. For example, we excluded attack goals such as *hardware reversal* due to Matter being mostly agnostic about the hardware, apart from generic requirements or recommendations such as the use of TRNGs or HSM. Based on the above attack goals, we derived the following threats, on a more technical level:

Table 4.8: Threat Scenarios

ID	Name	Description	STRIDE
T.1	Leak of the RCAC or ICAC private key	The attacker obtains the private key of the RCAC or ICAC. Together with the corresponding RCAC or ICAC, the attacker might be able to spoof a controller within the fabric	STRIDE
T.2	Leak of the NOC private key	The attacker obtains the private key of the NOC and can spoof a non-controller node within the fabric	STRIDE
T.3	Modification of the device certificates or key material	The attacker alters or removes a certificate or key material to DoS a device temporarily (RCAC, ICAC, NOC) or permanently (DAC)	D
T.4	Modification of firmware	The attacker alters or removes the device firmware	TDE
T.5	Exfiltration of firmware	The attacker is able to extract the firmware	I
T.6	DoS of the OTA DFU	The attacker is able to prevent or interrupt the OTA DFU, causing a device to not deploy potentially security-critical updates	(T)D
T.7	Leak of sensitive user data	The attacker is able to read sensitive application data by sniffing the network communication	I
T.8	Leak of sensitive data (on a hardware-level)	The attacker is able to read sensitive application data by sniffing the communication lines between ICs	I
T.9	Counterfeiting (spoofing of a Matter-certified device, although the device is not certified)	The attacker tricks the controller into believing the device is Matter-certified in order to be used as a Matter device	S
T.10	Leak of the vendor PAI certificate and/or PAA and PAI key material	The attacker obtains the vendor certificate and private key used for write-access to the DCL, allowing them to propose new entries (e.g., add/revoke a certificate, add a software update, change an URI)	STRIDE
T.11	DoS of a commissioned node	The attacker performs a DoS on a node	D

Continued on next page

Table 4.8: Threat Scenarios (Continued)

ID	Name	Description	STRIDE
T.12	Leak of network credentials	The credentials used to connect to the underlying network used by Matter are leaked	STRIDE
T.13	Remote device control	The attacker gains control over a device	RIDE

Attack Graphs

Prior to release of the specification, we created attack graphs based on our pre-release concept of Matter. They are found in Appendix F. We used graphs instead of trees, because they allow to reuse items, which is favorable property that is not possible with trees. Note that the attack graphs have not been adjusted after our specification review due to time constraints and are therefore only included for informational purposes. Future work on attack graphs for Matter should focus on: Possible scenarios that lead to an attack goal, distinguishing possible and impossible scenarios or calculating the likelihood and/or feasibility using a metric, determining and displaying the information required to execute the scenario as well as what happens or which security assumptions apply if the attacker does not get the required information.

4.1.5 SECURITY AND PRIVACY REQUIREMENTS

In the following, we list requirements we deem substantial for secure-by-design IoT devices that respect user privacy.

Table 4.9: Security (S) and Privacy (P) Requirements

ID	Requirement	Description	Justification	S	P
SR.1	No exposure of network credentials	Network credentials and other sensitive cryptographic key material must not be exposed to the attacker, even when a commissioned Matter device is stolen	Stolen network credentials can expose vulnerable devices not facing the public Internet and can also allow to bypass firewalls	X	X

Continued on next page

Table 4.9: Security (S) and Privacy (P) Requirements (Continued)

ID	Requirement	Description	Justification	S	P
SR.2	Cryptographically secure communication between networked devices	The encryption techniques used must not expose the inter-device communication, hashes must be cryptographically secure and sufficient randomness must be used. If required, key and/or nonce rotation must be used	Network sniffing should not expose the content of device communication, nor should the attacker be able to fully or partially decrypt encrypted communication, be able to predict chosen randoms or misuse hashes, perform statistical attacks using fingerprinting to distinguish two distinct commands, e.g., by comparing packet lengths	X	X
SR.3	Mutual authentication	Any two Matter devices communicating in a Matter network must be mutually authenticated	Without proper authentication, an attacker could spoof other devices and elevate privileges	X	X
SR.4	No vulnerable dependencies	The Matter specification must not make mandatory or optional use of known vulnerable technologies at the time of release	Vulnerable dependencies pose a security risk, if Matter itself cannot prevent the inherited vulnerability	X	
SR.5	Security update policy	Matter must enforce a secure security-update policy	Matter devices can only be secure in the long run if manufacturers respond quickly to vulnerabilities and breaches and provide security updates in a secure manner	X	

Continued on next page

Table 4.9: Security (S) and Privacy (P) Requirements (Continued)

ID	Requirement	Description	Justification	S	P
SR.6	Availability	The attacker must not be able to DoS device operation, including the communication between them, or the DCL	If the attacker is able to DoS a device, he can cause substantial harm or damage, e.g., if he performs a DoS right after a door was unlocked to prevent the user from locking it, eventually allowing physical access to the facility	X	
SR.7	Secure decommissioning	The user must be able to decommission the device, so that any sensitive or user-specific data is removed from the device	The user must have easy means to factory-reset the device in such a way that any non-factory and non-firmware data is removed to avoid exposure of network credentials and user data	X	X
SR.8	Replay attacks	Devices must be secure against replay attacks	An attacker could try to catch and repeat an (encrypted) command previously sent by the user. Matter must be robust against replay attacks	X	
SR.9	Ability to withstand well-known attacks and exploits	The protocol must be robust against well-known attacks and exploits known as Common Vulnerabilities and Exposures (CVE), e.g., published here https://cve.mitre.org/	A secure-by-design protocol must protect against the most prominently known types of attacks. Even though this last requirement is practically very hard to fulfill, setting it as ambition will greatly help to design a robust and secure protocol	X	

Continued on next page

Table 4.9: Security (S) and Privacy (P) Requirements (Continued)

ID	Requirement	Description	Justification	S	P
SR.10	Data minimization	The Matter protocol should not expose more data than is necessary for operation to protect the users' privacy	General Data Protection Regulation (GDPR) requires enterprises to respect the privacy of their users		X
SR.11	Control over personal data	User's should be given control on how each individual data item may be collected, stored and used. This could be implemented via a rights and role management or ACL. For example, a user should be able to decide whether a door lock may access the user's location to automatically lock and unlock, either as opt-in or opt-out	General Data Protection Regulation (GDPR) requires enterprises to respect the privacy of their users		X

4.2 REVIEW OF THE MATTER SPECIFICATION

During the course of this thesis, we security-analyzed the Matter specification Version 1.0 [30]. In the following, we explain the security-related technical functions of Matter in more detail. This chapter is a summary of Matter's security architecture, which we extracted from the specification. We will discuss the implication, our evaluation and potential weaknesses in Chapter 5, apart from the cryptographic primitives, are discussed in this chapter.

4.2.1 CRYPTOGRAPHIC PRIMITIVES

Matter uses a mature set of cryptographic primitives. Because of this, we decided to shift the analysis of each primitive to the end of this section and instead focus on an in-depth technical description of the requirements on each primitive, in order to be able to evaluate the primitives in their composition in the last subsection of this section. Note that a full explanation of each of established cryptographic principles (e.g., PRNG, hash) is out-of-scope of this thesis. We refer to standard literature, if required.

Cryptographically Secure Pseudo Random Number Generator

A Deterministic Random Bit Generator (DRBG) is used as Pseudo Random Number Generator (PRNG) and must be implemented with either (1) CTR DRBG (with AES-CTR), (2) HMAC DRBG (with SHA-256 or SHA-512) or (3) Hash DRBG (with SHA-256 or SHA-512) as described in NIST 800-90A [9]: The seed comes from a TRNG with at least 256 bits of entropy [30, Chapter 3.1].

The DRBG is used to initialize message counters [30, Chapter 4.5.1.1], to generate random numbers during the PASE [30, Chapter 4.13.1.2], a resumption ID and random numbers during the CASE [30, Chapter 4.13.2.3], the epoch keys of operational groups in a fabric [30, Chapter 4.15.3], a challenge for the Message Counter Synchronization Request of the MCSP [30, Chapter 4.16.3.1], an attestation nonce for the DA Procedure [30, Chapter 6.2.3], and a random nonce for the Node Operational Certificate Signing Request (NOCSR) Procedure [30, Chapter 6.4.6.1] and is used during the generation of cryptographic keys [30, Chapter 13.6.1].

Cryptographically Secure True Random Number Generator

The True Random Number Generator (TRNG) is exclusively called by the DRBG and must have an entropy of at least 256 bits [30, Chapters 3.1, 3.2]. It *may* be implemented according to NIST 800-90B [87], but alternate implementations are allowed [30, Chapter 3.2]. The DRBG is used to provide a random seed for the DRBG.

Cryptographic Hash Function

SHA-256 is used as defined in FIPS 180-4 [59] with a length of 256 bits and a block length of 512 bits [30, Chapter 3.3]. The cryptographic hash function is used as the underlying hash function for the HMAC-SHA256 function [30, Chapter 3.4], ECDSA signature verification function [30, Chapter 3.5.3.2], and for SPAKE2+, in the latter of which it is used to compute the context [30, Chapter 3.10], and it is used to generate the transcript hashes of the Sigma2 Key (S2K), Sigma3 Key (S3K) and session encryption keys during the CASE [30, Chapter 4.13.2.6].

Keyed Hash Message Authentication Code

The Keyed Hash Message Authentication Code (HMAC) function is used as defined in FIPS 198-1 [60] with the hash function described above as the underlying hash function, and produces output of the same length as the underlying hash function [30, Chapter 3.4].

The HMAC-SHA256 function is used as the underlying Key Derivation Method (KDM) of the Key Derivation Function (KDF) [30, Chapter 3.8], as the underlying PBKDF2 algorithm of the PBKDF [30, Chapter 3.9], as the underlying HMAC for SPAKE2+, used to compute the confirmation tags [30, Chapter 3.10] and to generate the destination identifier using the Identity Protection Key (IPK) as key and the destination message as input during the CASE [30, Chapter 4.13.2.4].

Public Key Cryptography

The public key cryptography of Matter is based on ECC with the `secp256r1` curve, defined in FIPS 186-4 as `NIST P-256` [58], and also known as `prime256v1` in NIST 800-186 [16]. Public keys

are generated as specified in SEC1 [14] of the Standards for Efficient Cryptography [30, Chapter 3.5]. Signatures are created (and verified) using the signature (verification) function of ECDSA as specified in SEC1 and the cryptographic hash function defined in Section 4.2.1 [30, Chapters 3.5.3.1, 3.5.3.2]. Elliptic Curve Diffie-Hellman (ECDH) is used to compute shared secrets as specified in SEC1 [30, Chapter 3.5.4]. Certificates are used in `X.509 v3` and `X.509 v3 DER` format, and their validity is checked according to RFC 5280 [13], considering the notion of 'current time', which might not be up-to-date in some circumstances. ECC is used for SPAKE2+ during the PASE [30, Chapter 3.10] ECDH is used to calculate the Shared Secret during the CASE [30, Chapter 4.13.2.3]. ECDSA is used as the signature algorithm for all Matter certificates [30, Chapters 6.2.2, 6.5.5], the Certification Declaration (CD) [30, Chapter 6.3.1], Node Operational Certificate Signing Request (NOCSR) [30, Chapter 6.4.7] and for signature generation and validation during the CASE [30, Chapter 4.13.2.3].

Authenticated Encryption with Associated Data

Authenticated Encryption with Associated Data (AEAD) is used to ensure the confidentiality and integrity of the message payload in exchanges between nodes. It uses AES-CCM as the underlying function as defined in NIST 800-38C [36] with a length of 128 bits for the key of the block cipher and the MIC, and a nonce length of 13 bytes [30, Chapter 3.6]. The AEAD encryption function takes a symmetric key, the payload, associated data and a nonce as input, and outputs the cipher text with the appended MIC [30, Chapter 3.6.1], while the decryption function symmetrically inverts the function [30, Chapter 3.6.2]. The nonce is either fixed [30, Chapter 4.13.2.3] or deterministically derived using the security flags, the message counter and the source node ID. For unicast sessions that are CASE-secured, the source node ID is determined via the secure session context, while for PASE sessions, the unspecified node ID is used as source ID [30, Chapter 4.7.1.1]. The secure session context of unicast sessions stores the session type and role, local and peer session identifier, `I2RKey` and `R2IKey`, `SharedSecret`, local message counter, message reception state, local fabric index, peer node ID, resumption ID, `SessionTimestamp` and `ActiveTimestamp` and sleepy parameters (idle interval, active interval, peer active mode) [30, Chapter 4.12.2.1].

The message payload is the payload `P`, the security flag, message counter and source node ID plaintext fields form the nonce `N`, the Message Flags, session ID, security flag, message counter, source node ID, destination node and message extensions form the Authenticated Additional Data (AAD) `A`, and the key `K` is determined by the session ID. The parameters `K`, `P`, `A`, `N` together with the length of the MIC (which is 16 bytes) are passed on to AES-CCM. The output consists of the encrypted message payload and a Message Integrity Code (MIC) and replaces the original message payload. If the AEAD produces an error, the security processing is aborted. AEAD is used to secure the confidentiality and integrity of messages [30, Chapter 4.7.2]

Message Privacy

To protect the privacy of messages, AES-CTR is used as specified in NIST 800-38A [35] with a nonce length of 13 bytes and a key length of 128 bits [30, Chapter 3.7]. The counter generation function shall be used as specified in NIST 800-38C [36] [30, Chapter 3.7.1], which is a guideline for AES-CCM. Message privacy is also referred to as privacy obfuscation or privacy encryption, and the privacy key is derived using the KDF described in Section 4.2.1 with the encryption key as input

key, an empty salt and a target output length of 128 bits [30, Chapter 4.8.1]. The privacy nonce is the concatenation of the session ID and the lower 11 bytes of the MIC produced by the AEAD encryption [30, Chapter 4.8.3, Figure 10], both of which are known to the attacker in the Dolev-Yao model. Therefore, the privacy nonce can be considered known to an attacker.

The privacy key is the key `K`, the privacy nonce the nonce `N`, and the message header fields message counter, source node ID, destination node ID and message extensions are the message `M`. These parameters are passed on to AES-CTR, where the message `M` is the payload `P`. The output replaces the message header fields that formed the message `M`. The message flags, session ID and security flags remain unprotected. Message privacy is used to protect the message counter and source node ID, the destination ID and message extensions [30, Chapter 4.8.3]. The KDFs in Matter use a *FixedInfo* [30, Chapter 3.8].

Key Derivation Function

The Key Derivation Function (KDF) used in Matter uses an HMAC as specified in NIST 800-56C [8]. It takes a key, salt, informative string and desired output length as a multiple of 8 bits as input and returns a bit array. It is allowed to generate multiple concatenated keys using a single KDF call [30, Chapter 3.8]. The KDF is used to generate encryption keys during the PASE [30, Chapter 3.10].

Password-Based Key Derivation Function The Password-Based Key Derivation Function in Matter uses an HMAC as specified in NIST 800-132 [86] which takes a passcode, a salt between 16 and 32 bytes, the number of iterations (between 1,000 and 100,000) and length as input, and returns a bit array. The passcode maps to the key, and the rest of the input maps to the payload of the underlying HMAC function. The number of iterations and salt are part of the the PBKDF parameter exchange (`pbkdf_parameters`) between a commissioner, also referred to as prover or initiator, and a commissionee, also described as verifier or responder in the context of PBKDF, during the PASE and determined by the commissionee [30, Chapter 3.9]. The PBKDF is used to derive a strong secret from a weak passcode. During the PASE, strong one-time keys are negotiated with distinct keys for each direction [30, Chapter 4.7.1]. The PBKDF is used to derive PAKE values for the initiator and responder during the PASE and a shared secret during the PASE.

Password Authenticated Key Exchange

The Password Authenticated Key Exchange (PAKE) is specified as SPAKE2+ [82] [83] [30, Chapter 3.10] (see Section 2.3.2). The points `M` and `N` of the elliptic curve used in Matter are correctly taken from the SPAKE2+ specification draft. Matter uses elements of both version 2 and 8 of the SPAKE2+ specification as described in the analysis below. The PAKE values of the commissionee are calculated at manufacturing time, which is only possible because SPAKE2+ is used without including identity information of the two communicating parties. The SPAKE2+ specification draft warns about Unknown Key Share (UKS) attacks [11], that result from non-implicit identities, as further discussed in the following section. The PAKE is used to derive a shared secret during the PASE.

4.2.2 SECURITY ANALYIS OF THE CRYPTOGRAPHIC PRIMITIVES

Random Number Generators

The choice of algorithms for the cryptographic primitives follows national guidelines such as those published by National Institute of Standards and Technology (NIST), which are commonly recommended by security experts and considered best practice. Using a TRNG as a seed for the PRNG in form of a DRBG is good practice to eliminate the downsides of both TRNGs (e.g., physical wear-off, high costs) and PRNGs (e.g., predictability for known seeds). TRNG and PRNG both have their distinct strengths and weaknesses. PRNGs suffer from statistical predictability and deterministic behavior for known seeds, while TRNGs rely on the randomness of physical phenomena which makes them less cost efficient and subject to gradual decline. Combining the two can limit the negative effects and result in a good approximation of randomness. However, it is left to the manufacturer to choose and implement a secure TRNG, as he is not required to follow NIST 800-90B [87]. The only requirement on the TRNG is its entropy, but there are no requirements on the statistical properties, randomness, whether the entropy source is digital or analog [10], and whether a stochastic model is available, that allows to determine proper configuration and determine erratic (not truly random) results, along with a proof of quality.

Hashes, HMACs and KDFs

The SHA-256 algorithm used to calculate a cryptographically secure hash during the PASE and CASE is vulnerable against length extension attacks, where the attacker is able to calculate a valid hash for an attacker-controlled `message1||message2`, where `message2` is appended to the original `message1` of the victim. It is used to hash the context or received messages as part of the shared secret's input. The vulnerability does not weaken Matter's security, because any modification of the hash value results in a difference in the shared secret. During the PASE it is detected due to the difference in the confirmation tags `cA, cB`, and during the CASE it is detected when the responder receives the first encrypted message, which he will fail to decrypt.

HMACs were specifically designed to mitigate these attacks, and to the best of our knowledge, no length extension attacks are known against the current HMAC specification, in which the MAC is generated the following way: `Hash(key||Hash(key||message))`. Additionally, they are less prone to collisions and do not necessarily inherit the weakness of their underlying hash functions, even though their security still depends on them. HMAC can be broken by brute-force if the key is weak, for example when the key is too short or predictable. When the key is longer than the size of the hash block (here: 512 bits), `HMAC(Hash(key), message)` is calculated instead of `HMAC(key, message)`, which was shown to be susceptible against collision in some scenarios. Our Matter specification review revealed that the key used for HMAC is either the passcode, the key confirmation keys `KcA`, `KcB` and the IPK, none of which exceed 512 bits, or the salt used for the KDF, which is either extracted from the `PBKDFParameterSet` (128 to 256 bit), is the target's operational fabric ID (64 bit), a concatenation of the IPK (16 bit), responder random (256 bit), responder

10 Analog entropy sources are generally more vulnerable against Electromagnetic Interference (EMI)-based attacks, but both can produce high-quality output.

ephemeral public key (520 bit) and transcript hash (256 bits) for the calculation of the S2K (key used for the second SIGMA message during the CASE), a concatenation of the IPK (16 bit) and transcript hash (256 bits), a concatenation of the initiator random (256 bit) and resumption ID (128 bit) or the rotation counter (16 bit). The salt of the KDF function used as the key of the HMAC exceeds 512 bits during the calculation of the S2K, where the HMAC function is the auxiliary function used by the KDF function. Still, there is no scenario where an HMAC-based attack can be exploited, because the HMAC is used to generate a shared secret where the victim would abort the protocol as he would not be able to properly decrypt the attacker's message.

Public Key Cryptography

Public key cryptography and ECC are used according to the security guideline SEC1 [14] and RFC 5280 [13]. We acknowledge the analysis of ECC and PKI this as a limitation of the thesis (see Section 5.27). In 2012, Heninger et al. in 2012 [45] described how public key cryptography becomes vulnerable if unverified RNGs are used and found that the majority of problems appeared in headless or embedded devices, which are a subgroup of IoT. It is important for manufacturers to understand, that the entire security of Matter depends on the correct implementation of a good randomness, else Matter devices risk reusing specific randoms more often than others, which might be exploited by an adversary. For the following considerations, we assume Matter's public key cryptography to be secure.

AEAD and Message Privacy

AES-CCM is used for message encryption, and AES-CTR for privacy encryption. The AEAD used during the CASE uses fixed nonces for the `Sigma2` message. For message encryption, the nonce, also referred to as IV, is derived from the security flags, message counter and source node ID [30, Chapter 4.8.3, Figure 10], and for privacy encryption, the nonce is derived from the session ID and lower 11 bytes of the MIC. We derive that for both message and privacy encryption, nonces are either known or derivable by the attacker. Using a fixed nonce for the `Sigma2` message is acceptable, because subsequent `Sigma2` messages use fresh random keys. Due to the high message and key entropy, precomputation is not feasible, and the 128-bit key length allows to securely use a fixed nonce for the `Sigma2` message. For message and privacy encryption, where messages could be sent using the same key, we consider the following: AEAD uses AES-CCM as the underlying block cipher in stream cipher mode, as described in Section 2.3.1. Stream ciphers are generally vulnerable to three types of attacks: bit-flipping attacks, chosen-IV attacks and reused key attacks. Andreeva et al. [4] argue that the security of AES-CCM is practically reduced to AES-CTR if the authentication is not properly verified prior to release of the plaintext to memory, the Releasing Univerified Plaintext (RUP) setting, because the attacker is not required to forge a ciphertext in order to obtain information about the corresponding (unverified) plaintext. Because the authentication is temporarily bypassed in the RUP setting, the devices become vulnerable against modification of cipher text such as bit-flip attacks. This might occur when decryption and verification of AES-CCM are interleaved and the encrypted or decrypted payload is stored in Random-Access Memory (RAM) or persistent storage. Matter defines a single function for decryption and verification [30, Chapter 3.6.2] which returns the verification result and specifies, that further processing shall fail if verification

has failed [30, Chapter 4.7.3]. Therefore, we do not need to consider attacks resulting from partially decrypted, yet-to-be verified payload that is further processed. However, because Matter does not forbid to store the plaintext or ciphertext in RAM or persistent storage during the function, the process risks that the attacker is able to store malicious payload on the device for a short period of time, or target a third person by storing sensitive information. To exploit this, the attacker would additionally be required to find a Remote Code Execution (RCE) vulnerability suitable to invoke or move the temporarily stored code. AEAD used in this way is still *more* secure than unauthenticated encryption methods, but to leverage the full advantages of the scheme, Matter could add additional requirements on the handling of unverified data, that originated from a receive buffer. Another possible weakness of AES-CCM is that it mitigates, but does not fully prevent, reused key attacks and bit-flipping attacks. Since Matter explicitly describes rekeying for situations where the message counter wraps (see Section 4.2.14), the risk of nonce collision for message encryption is negligible, and therefore reused key attacks against message encryption are a negligible risk. The privacy nonce is more likely to collide because it only requires two messages `m1`, `m2`, that share the same lower 11 bytes of a MIC for the same session. In the rare event of privacy nonce collision, the message counter, source code ID, destination ID and message extensions are revealed, from which the encryption nonce can be derived. The event would slightly reduce privacy, but does not affect message encryption. In the unlikely event that a key for one of the SIGMA messages was reused during the CASE, the plaintext of exactly one SIGMA exchange message would be revealed. The attacker could obtain public keys, certificates, SED parameters, signatures and a resumption ID, which impacts privacy. Because the shared secret is derived from the private and public keys of the two communicating parties, the attacker is not able to derive it, and can therefore not perform a MitM attack on the SIGMA exchange unless he manages to break Matter's public key cryptography, even if the entire exchange was unencrypted. To sum up, breaking the encryption of the SIGMA exchange can reduce privacy, but not security.

If the attacker used a bit-flipping attack against the privacy encryption, he could bit-flip either the message counter, source or destination node ID or the message extensions. However, their values are used as the additional data for the AEAD and therefore, any manipulation of these fields would cause an authentication failure on receipt, as long as the used cipher is secure. We derive that the protocol is secure against modifications of any of the Matter message fields sent in plaintext or privacy-encrypted. Similarly, bit-flipping against the message encryption causes an authentication failure. Since the MAC is calculated using the message payload, it is not possible to reject bad ciphers before they are completely decrypted, which in theory could be abused during a DoS attack by an attacker that is within the operational network and initiates communication. There is a remaining risk of chosen-IV attacks, which allow to infer information about the secret key or plaintext, when chosen IVs produce non-random patterns of cipher text, which we could not exclude. Note that for CCM, the IV used in CBC-MAC must not collide with the counter values used by the CTR mode and must be unique for every message.

Privacy and message encryption use two distinct keys. The privacy key is deterministically derived from the encryption key using an unsalted KDF. Therefore, privacy encryption is at most as strong as the message encryption, as it fully relies on the secrecy of the encryption key, and any attacker in possession of the encryption key is able to decrypt privacy encrypted messages. The attacker cannot decrypt message encryption if he is in possession of the privacy key, but not of

the encryption key, because the KDF used to derive the key is based on an HMAC and therefore a cryptographic one-way function. For privacy encryption, the counter generation of AES-CTR uses a recommendation for AES-CCM [30, Chapter 3.7] [36]. Since the fields used to derive the message encryption nonce are encrypted, the privacy encryption also protects against attacks targeting the CBC-MAC mode of AES-CCM, for which knowledge of the nonce is required [11]. The *FixedInfo* [10, Chapter 5.8.2] binds the derived key material to the context of the transaction, for which human-readable labels are used in Matter, that describe the key context. By this, attacks that are based on confusion about the context of the key derivation are prevented [10, Appendix B].

PAKE and SPAKE2+

Matter uses SPAKE2+ as its PAKE algorithm, as discussed in Section 2.3.2 and Section 4.2.1. A PAKE protocol allows to derive a common key over an insecure channel, using a low-entropy password. The execution of the SPAKE2+ protocol during the PASE is secured by BLE, the operational network (e.g. Wi-Fi, Thread) or Wi-Fi Soft-AP depending on device capabilities and whether the device is already within the operational network [30, Chapter 4.3.1]. A commissionee can only enter the PASE if it was factory reset or set in commissioning mode [30, Chapter 5.5.1]. Because of this, SPAKE2+ is usually performed in a secure context, but this is not guaranteed, e.g., because Matter supports operation over unsecured channels. Note that a consideration of weaknesses and preliminaries required to break security of the underlying communication channel are out-of-scope for this thesis, but we provide constituted recommendations in Section 4.2.9. In the following, we will assume that SPAKE2+ is conducted over an unsecured channel. Matter uses version 2 of the SPAKE2+ draft [82]. In May 2022, version 8 of SPAKE2+ draft has been released [83]. Because Matter's implementation of SPAKE2+ includes parts of both version 2 and 8 of the SPAKE2+ draft, we provide the most relevant changes: Some ambiguities were dissolved and some underspecifications addressed. The preamble phase, or offline registration, has been included as in-scope of the specification, that explicitly describes how to handle absent identities, which is equivalent to Matter's implementation in this aspect. Furthermore it has been clarified that the choice of the PBKDF for the computation of the distributed values $\mathrm{w0}$ and $\mathrm{w1}$ is out-of-scope as it does not affect interoperability. Additionally, it has been added that the ephemeral randomness, probably referring to x and y, must be generated using a a cryptographically secure PRNG. Note that instead of a PRNG, Matter uses a TRNG that seeds a DRBG, which is a PRNG. This is an acceptable deviation, due to the security considerations explained at the beginning of this subsection. Starting from version 5 of the specification draft, the importance of group member checks is emphasized, how it is done and that a failure to do so implicates subgroup confinement attacks. To the best of our knowledge, Matter has not implemented any group membership checks. Matter correctly adopts the two random elements (compressed points) `M` and `N` as suggested in the SPAKE2 [52] and SPAKE2+ specification, for which it is known that their discrete logarithms are unknown[30, Chapter 3.10]. The newest version includes minimum output and key lengths, such that for P-256, the output of the PBKDF (here: $\mathrm{w0s} \,||\, \mathrm{w1s}$) should be at least 640 bits (80 bytes). The requirement is fulfilled by Matter since it uses 40-byte keys, that are concatenated. Because Matter mixes version 2 and 8, we provide an overview of SPAKE2+ renaming make it easier to follow the below evaluation:

11 An example of such an attack is the precomputation attack described in RFC 3610 [93].

identity A → idProver

identity B → idVerifier

public share pA → shareP

public share pB → shareV

confirmation tag cA → confirmP

confirmation tag cB → confirmV

shared secret (confirmation) Ka → K_main *

shared secret (encryption) Ke → K_shared *

confirmation key KcA → K_confirmP *

confirmation key KcB → K_confirmV *

Calculations have been adjusted (*) for unspecified reasons:

$$\begin{aligned} \text{K_main} &= \text{hash}(\text{TT}) \\ \text{K_confirmP} \parallel \text{K_confirmV} &= \text{KDF}(\text{nil}, \text{K_main}, \text{“ConfirmationKeys”}) \\ \text{K_shared} &= \text{KDF}(\text{nil}, \text{K_main}, \text{“SharedKey”}) \\ \\ \text{confirmP} &= \text{mac}(\text{K_confirmP}, \text{shareV}) \\ \text{confirmV} &= \text{mac}(\text{K_confirmV}, \text{shareP}) \end{aligned}$$

Matter uses Ka, Ke, KcA and KcB of draft version 2, but the calculation of Matter's confirm values cA, cB equals the equations of confirmP and confirmV of draft version 8. The key difference is, that the full hash of the protocol transcript TT instead of half the hash is used to derive the confirmation keys, and the KDF is used to derive the shared encryption key from the hash. This is probably done to prevent attacks based on collisions of the first half of the hash, which could result in an attacker being able to produce the same confirmation keys for different contexts, which would make the protocol vulnerable to MitM attacks. In Matter, the PBKDF parameter request and response are a part of the protocol transcript TT, because they are part of the Context.

The PBKDF parameter exchange introduces additional randoms, one 256-bit random each for each of the communicating parties, used to mitigate offline precomputation as well as UKS attacks. An Unknown Key Share (UKS) attack occurs when the attacker E establishes one connection each to the parties A and B, where A assumes to be connected to either of the parties A or E, and B assumes to be connected to A. In this scenario, E uses the identity fingerprint of party A to trick B into believing it shares its key with A, although A and B did not directly connect to each other. The attack implies spoofing and is similar to a MitM attack, but with the difference that the party from which the identity is spoofed may be aware of its communication with the adversary, while not suspecting an attack. From an attacker side, the values of the PBKDF parameter request can be set arbitrarily

within reasonable limits, because the commissionee does not perform plausibility checks. Therefore, the attacker controls the initiator random, unless he performs a MitM or UKS attack. The attacker is able to fully eliminate one of the randoms by always reusing the same random for any connection attempt, instead of producing a random of their own. Still, it can be argued that one of the randoms alone sufficiently mitigates offline precomputation under the assumption of cryptographically secure RNGs in use by the genuine party, when we assume the following (maximum) entropy for one genuine party in the exchange: $(\text{entropy of random}) \cdot (\text{entropy of passcode}) \approx (2^{256}) \cdot (2^{26.575}) > 1.157e85$ (see Section 4.2.4 for an explanation of passcode entropy). Because both parties produce a random that contextualizes the key exchange, weak keys can only be derived when both implementations are flawed. The two randoms reduce the risk of UKS attacks to a negligible level because two distinct parties A and B are highly unlikely to respond with the same random, which results in the attacker usually not being able to share the same key with both parties. UKS attacks are mitigated to a negligible level for similar reasons. The attacker is not able to derive `V` and can therefore not calculate the shared secret unless he is in possession of the passcode or PAKE values. However, the risk of offline precomputation attacks ia only reduced and not eliminated, because there is a risk that the manufacturer implements a weak RNG, which can significantly reduce the computational and storage costs. An attacker that is able to break the PAKE via an offline precomputation will be able to either sniff network credentials *or* take administrative control during one instance of the attack, because he must decide between passive eavesdropping and active pairing.

The attacker has greater chances to successfully launch an UKS or offline precomputation attack if the RNG used by the commissionee is flawed and the attacker can use subsequent commissioning attempts to influence the choice of the commissionee's random to their benefit. In Section 4.2.15 we describe how the attacker can abuse status report messages to trigger the commissioner to send another random. The attack also depends on the entropy of the password pw, which in Matter is the 27-bit setup code. We show in Section 4.2.4, that the entropy approximates $2^{26.575}$ instead of 2^{27}. During the offline initialization, or preamble, as defined in Matter, w0 and w1 are distributed to both devices, and therefore, the attacker can perform a lookup and try to derive the setup code. While online attacks can be interrupted when discovering failed attempts, it is hard to prevent or mitigate offline attacks. If the setup code is static, it is sufficient for the attacker to perform the offline precomputation and subsequently force the commissionee to commissioning mode to perform the attack. In order to launch an offline precomputation attack, the attacker would be required to observe the commissioning process of two devices. Online precomputation attacks are significantly harder, especially when dynamic passcodes are used for commissioning, because the PASE timeouts after 60 seconds, and commissioning mode is exited by nodes after 20 failed attempts [30, Chapter 5.5]. Therefore, it is reasonable to differentiate between brute force attacks where the attacker can enable commissioning mode and those where they cannot [30, Chapter 13.3].

Matter implements an outdated version of the derivation of shared secrets, where the hash of the protocol transcript is split in half for Ka and Ke. This can potentially cause vulnerabilities, because the properties of a cryptographic hash [12] are not inherited by its split halves. This is in so far an interesting change as that in draft version 2, the length of the confirmation keys is quarter the size of $\text{hash}(\text{TT})$, but in draft version 8 it is half the size of it, i.e., 64 bits versus 128 bits for P-256

12 Cryptographic hashes fulfill the properties pre-image resistance, second pre-image resistance and collision resistance.

with SHA-256. The adjustment in length is made at cost of an additional KDF computation. We are not sure why Matter did not apply the fix introduced in SPAKE2+ draft version 8, but assume it may have been done out of economic reasons, as we believe it can be easily argued that the the computations in draft version 8 are at least as secure as the computations in draft version 2.

The PASE results in two unidirectional keys `I2RKey` and `R2IKey` and an attestation challenge that is used for the DA Procedure, which are derived from the shared secret Ke, using a KDF without reduction of key entropy [30, Chapter 4.13.1.2]. Using unidirectional keys limits the attacker to being able to expose their own messages, but not those of a genuine party, while the attestation challenge binds the device attestation query to the DCL to a specific instance of a PASE without revealing any of its contents, maximizing privacy and providing replay attack prevention for device attestation queries. Overall, and despite our considerations of potential attacks, we argue that Matter implements a strong PAKE, because of the high time effort and resources, and preliminary conditions required by an attacker to launch the attack. This greatly limits the set of potential attackers in so far as that we assume that the risk is reduced to targeted attacks, which we consider a very beneficial feature of the pairing. We translated both SPAKE2+ and Matter's use of SPAKE2+ in the PASE context to symbolic models in form of Tamarin theories, see Section 4.3 for more details.

4.2.3 USE OF UNSECURED COMMUNICATION

Unsecured communication is strictly limited to device discovery, initiation of the commissioning process using User Directed Commissioning (UDC) and protocols which bootstrap secure messaging such as PASE or CASE [30, Chapters 4.1.1, 4.4]. Furthermore, *secure channel status reports* used during session establishments necessarily need to be unencrypted [30, Chapters 4.12.1]. This seems to be a reasonable limitation with respect to the PKI architecture in use by Matter (see Section 3.5.2), and highly limits what information can be exchanged over unsecured messages.

Any unencrypted communication is an entry point and potential attack surface in Matter because of its indirect claim to be secure in the Dolev-Yao model [13]. Device discovery delivers device addresses and parameters, UDC acts as a 'doorbell' function, session establishments are used to derive shared secrets and status reports signal operation results. We evaluate the security of device discovery and UDC in Section 4.2.6, session establishments in Section 4.2.13 and status reports in Section 4.2.15.

4.2.4 DEVICE PASSCODE AND ONBOARDING PAYLOAD

The passcode has a length of 8 bytes [30, Chapter 3.10] and stores a 27-bit secret encoding an 8-digit decimal numeric value. It is used as a shared secret during the PASE. Since there are about $10^8 = 100,000,000$ valid passcodes, the entropy of the passcode is in the order of approximately $2^{26.575}$, which contradicts CM5 of the Matter specification and other specification parts, which state that the setup code has at least or equals 27 bits of entropy, which would require at least $134,217,728$

13 Matter claims to be secure independent of the underlying communication technologies, therefore we may assume that the attacker is within the network and can arbitrarily eavesdrop on unencrypted communication as well as inject messages if we want to refute or verify the claim.

possible passcodes. This implies a confusion between the terms 'key length' and 'entropy' by the authors of the specification. Generally, it is not good practice to round up when it comes to entropy, because the rounding can cause follow up errors in security considerations. Eleven trivial passcodes such as 00000000 are not allowed to be used [30, Chapter 5.1.7], possibly because they can be expected to be included in dictionaries of Non-Matter-specific cracking tools.

Passcodes are either static, dynamic or ephemeral and manufacturer-provided. Static passcodes are required to be random on an individual device basis and the device must be supplied with a PAKE passcode verifier in its internal storage, the passcode must not be accessible during and via the operational mode, and the passcode must be stored and processed physically isolated from the PAKE passcode verifier. If a passcode is derived, it must not be derivable from public information and must use a cryptographically secure RNG. Furthermore, any dynamically derived passcodes may be accessible to the commissioner only during the commissioning process [30, Chapter 5.1.7]. If the attacker is able to obtain the static passcode, for example by having one-time physical access to the device with passcode on-device, he can eavesdrop on device commissioning or perform a MitM attack on it to manipulate device configuration, e.g., provide compromised network credentials [30, Chapter 5.2.2.1.3].

Ephemeral passcodes are distributed in the custom commissioning flow [30, Chapter 5.7.3.2.1], see Section 4.2.8. It is unclear whether ephemeral and dynamic passcodes are the same, but we assume them to be distinct concepts, because devices can generate dynamic passcodes [30, Chapter 5.1.7]. Dynamic passcodes are not explicitly described in Matter, therefore we could not analyze their security. We assume that the passcode ID corresponding to the PAKE passcode verifier currently being set to 0 implies that dynamic passcodes are meant to be introduced in future versions of the specification, with a touch of doubt [30, Chapter 6.6.2.1]. It therefore seems possible for Matter devices to allow rotation of the passcode such that each commissioning of a device uses a fresh passcode the attacker cannot know beforehand, either with (ephemeral) or without (dynamic) connection to the internet.

The passcode is included within the onboarding payload in form of a QR code, NFC tag [30, Chapter 5.1.8] or a manually entered code. For example, if the onboarding payload comes in form of a QR code or NFC tag, it has to include a 1- or 3-bit version of the payload, the VID and PID to identify the make and model of the device, required for DA and OTA DFU. Furthermore, it includes 2 bits specifying the device commissioning flow, which indicates what steps are needed prior to commissioning that may require interaction with the DCL, the discovery capabilities bitmask indicating how the device can be discovered and a 12-bit discriminator value to quickly distinguish advertising devices and the passcode. Furthermore, a 4-bit padding and byte-aligned variable size Tag Length Value (TLV) data may be included to provide additional machine-readable information, such as salt and number of iterations used for the PBKDF [30, Chapters 5.1.1, 5.1.5.2]. A single QR code may encode multiple concatenated onboarding payloads to allow a single QR code to onboard multiple devices. Commissioners are not required to support concatenated onboarding payloads, but if they do, they are required to onboard the devices in the exact same order in which the onboarding payload is concatenated [30, Chapter 5.1.6].

An NFC tag should only be used during commissioning, if the onboarding payload is attached to the device permanently, otherwise the attacker could swap the tags of two devices. Furthermore, it must be protected from read-out in closed packaging and not be alterable [30, Chapter 13]. Any

failures to do so allow an attacker to maliciously onboard the target device, trick the user into commissioning with an attacker-controlled device, or perform a DoS attack on the commissioning of the device [14]. The countermeasure requires manufacturers to either shield the NFC tag from read-out, e.g., by attaching an Radio Frequency Identification (RFID) NFC signal blocker, or use packaging that prevents the attacker from reaching sufficient proximity [15]. The manual pairing code is a trimmed, human-readable version of the onboarding payload [30, Chapter 5.1.4].

Using salt and iterations is a common practice to significantly increase the computational costs of a rainbow and dictionary precomputation attack, and does not significantly impact on the costs of a brute-force attack. The disclosure of the salt and number of iterations is generally considered secure. It is notable though, that knowledge of passcode, salt and number of iterations in combination reduces the computational costs to pre-compute a rainbow table of PAKE values for the SPAKE2+ exchange.

Manufacturers may place the onboarding payload on-device or in-packaging. If it is on-device, it may be removable or obscured, shown via the device display or presented as audio output, for example. It might be hidden inside the battery compartment cover or even behind a tamper-monitored cover, among other possibilities [30, Chapter 5.7.4]. CM89 recommends to attach the code such that it is not photographable [30, Chapter 13.6.2], because this prevents attacks in which a long-range camera or similar means are used to obtain the passcode.

For multi-function devices such as a Matter node on a smartphone which is also used for the administration app, the commissioner should implement a copy-and-paste mechanism to allow copying the passcode from one application into the other [30, Chapter 5.1.4.2]. Without further restrictions, this risks passcode leakage, which we elaborate in Chapter 5.

A device passcode can be recovered by the manufacturer's customer support using the rotating device identifier, which is optionally advertised during commissioning [30, Chapter 5.4.2.4.5]. It must be irreversible, such that it does not allow recovery of the unique identifier of the device by parties without prior access to the set of possible device identifiers, and the leakage of a common key must not lead to recovery of devices sharing the key. It must protect against long-term tracking by rotating upon commencement of advertising [30, Chapter 5.4.2.2], must have at least 64 bits of entropy and must not contain fixed identifiers. The passcode security relies on the secrecy of the unique identifier. Recovery requires a proof of possession, which is out-of-scope of the specification [30, Chapter 5.4.2.4.5]. We discuss the implications of passcode recovery in Chapter 5.

4.2.5 CURRENT TIME AND CERTIFICATE VALIDATION

Correct time synchronization is not self-evident in distributed systems and impacts the handling of certificate validity and revocations. Matter uses UTC, offers multiple methods for time synchronization and uses the concept of *Last Known Good UTC Time* as a fallback, which out-of-box is the

14 Note that the DoS is potentially permanently, and although there is a passcode recovery mechanism, Matter has claimed that devices remain usable even if the company ceases to exist, therefore, security measures should not involve manufacturer actions.

15 Unlike for BLE, for which the attacker can significantly increase ranges, NFC range extensions rather focus on relay techniques, instead, and the relay would require equal proximity as the attacker. This difference is due to how load modulation is used to transmit data, and therefore, proximity-based protective measures are acceptable for NFC.

firmware compile-time [30, Chapter 11.16]. Nodes are recommended to store a Last Known Good UTC value at least once a month [30, Chapter 3.5.6]. When a node switches from the Last Known Good UTC Time to a synchronized time, it should recheck security material and existing connections [30, Chapter 3.5.6.1]. Nodes may consider a certificate path valid even though the validity period of a certificate is invalid [30, Chapter 3.5.6]. It is underspecified how revocation information is obtained from the DCL. The commissioner can configure an Network Time Protocol (NTP) server on a node for time synchronization purposes. Nodes with an always-on power source should act as NTP servers to the Matter network [30, Chapter 11.16]. Nodes that synchronize using an external source are recommended to use Network Time Security (NTS), if it already has a TCP/TLS stack and attempt time synchronization before using potentially expired security material. Global Navigation Satellite System (GNSS) should be preferred over NTP, which gain is preferred over synchronization with other nodes.

4.2.6 SECURITY OF DEVICE DISCOVERY

As described in Section 3.3.1 and Section 3.3.4, DNS-Based Service Discovery (DNS-SD)-based [19] device discovery is used to discover devices for commissioning. The discovery information consists of the discriminator and optionally VID, PID and extended data, the latter (if present) contains the rotating device identifier that contains information about the factory software version and related features, manufacturing date or setup code recovery (see Section 4.2.4). We investigated whether DNS messages can contain user-controlled input and found that the 32 bytes `TXT` record for the Distinguished Name (DN), that is used for the device name and optionally advertised, is editable by the user. Target hostnames are derived from the device MAC address, and when the host performs MAC address randomization, the randomized version is advertised. Thread meshes play a special role during service discovery, because Thread Border Routers (or capable devices) offer discovery service proxies to deliver the multicast advertisement of Matter nodes into the Thread mesh [30, Chapter 4.3].

Discovery is divided into operational discovery, commissionable node discovery and operational discovery. Operational discovery offers a function that allows to enumerate non-specific services for diagnostic purposes [30, Chapter 4.3.2.6]. Other than not-yet commissioned devices, already commissioned devices are not required to be in commissioning mode. The commissionable node discovery is performed by sending a broadcast message, and when the user selects the commissionee, its commissionable node discovery DNS-SD `TXT` record provides further guidance on how the user can put the commissionee in commissioning mode [30, Chapter 5.2.2.3]. Commissioner Discovery can be initiated by on-network devices that wish to be commissioned. A use case example is a television, which the user wants to control from his smartphone [30, Chapter 4.3.3]. For this, the commissionee may use the User Directed Commissioning (UDC) 'doorbell' function by which it signal to the commissioner that it wants to be commissioned, upon which the commissioner prompts the user for confirmation. The commissioning process is started when the user enters the onboarding payload into the commissioner [30, Chapter 5.2.2.4]. Matter has no trust mechanisms to identify trustworthy commissioners [30, Chapter 5.2.2.4.3], which strengthens the assumption of rogue commissioner attacks, which we introduce in Section 4.2.7.

4.2.7 SECURITY OF DEVICE COMMISSIONING

When commissionees use commissioner discovery to initiate the commissioning process, the user interface (if present) may display the returned information such as the VID, PID, friendly name, discriminator and extended data. may be displayed to the user[30, Chapter 4.3.3]. During commissioning, commissioners establish the authenticity of commissionees as certified Matter devices in the operational network. Even if a commissionee fails the DA, it may still be commissioned to the Matter network, and it is recommended to warn the user [30, Chapter 5.5].

If the commissioner proceeds commissioning after DA procedure, it request the commissionee to generate a new operational key pair and eventually installs the NOC on the commissionee, granting access to the fabric. After this, the commissioner may configure the commissionee's ACL and operational network, if possible and required. The commissioner triggers the commissionee to connect to the operational network, unless the commissionee is already connected to it [30, Chapter 2.8], after 20 failed attempts commissioning is aborted [30, Chapter 5.5.1]. Commissioners may learn about networks that are visible to the commissionee [30, Chapter 5.5].

Out-of-box commissioning is divided into three flows: standard, user-intent, and custom. The standard flow requires no device interaction apart from applying power, the user-intent flow requires user interaction, and the custom flow requires additional interaction by the user and guidance of the manufacturer. Both standard and user-intent flows are available for initial commissioning [30, Chapters 5.7.1, 5.7.2], while the custom commissioning flow is guided by a manufacturer service [30, Chapter 5.7.3]. The major difference between the standard and user-intent flow is, that the standard flow device starts advertising upon power-application, while the user-intent flow device will wait for the user to trigger advertisement. The attacker can assume that the standard flow is available if he has obtained a manual pairing code without PID and VID [30, Chapter 5.1.4.1] The custom commissioning flow involves a `CommissioningCustomFlowUrl` field, which is published via the DCL and guides the commissioning [30, Chapter 5.1.1.3]. During the custom flow, the manufacturer can provide fresh ephemeral passcodes, see Section 4.2.8. As a side remark, the use of the custom flow contradicts Matter's claim, that the devices can still be used even if a manufacturer ceases to exist [16]. A fail-safe can be used during the commissioning window by creating a fail-safe context on the commissionee, that is used to track, among others, the network configurations, addition and updates of NOCs and the operational credentials. If the fail-safe expires before commissioning is complete, the changes that were made during commissioning are reverted [30, Chapter 11.9.7.2].

4.2.8 URIS PROVIDED BY THE DCL

Manufacturers can provide links such as Uniform Resource Identifier (URI)s or Uniform Resource Locator (URL)s in the DCL for various purposes, such as the custom commissioning flow [30, Chapter 5.7.3] or OTA DFU [30, Chapter 11.19.2]. HTTP-based URLs are required to use HTTPS [30, Chapter 5.7]. The URL parameter `MTop` is reserved to transmit operational passcodes and should

16 Various developers of Matter had previously claimed, that Matter is fully operable without the cloud. This is therefore only true for a subset of devices, and not guaranteed.

not be used to transmit a passcode on-device or in-packaging, but returns a dynamic, ephemeral passcode which the manufacturer provides to the user during the custom commissioning flow [30, Chapter 5.7.3.2.1]. There is a risk, that a device accidentally submits a static passcode, which must then be considered known against the most powerful malicious actors as described above as well as the manufacturer.

4.2.9 UNDERLYING COMMUNICATION CHANNELS

As explained in the previous section, the commissioning is secured by either BLE, the operational network (e.g. Wi-Fi, Thread) or Wi-Fi Soft-AP [30, Chapter 4.3.1]. The underlying channel influences the possibility and feasibility of attacks against commissioning and operational communication. Neither does it directly expose cryptographic keys, nor does it allow administrative control by its own. Matter chooses the most secure protocol based on beacons and probe responses [30, Chapter 11.8.8.5]. Networks are uniquely identified by the SSID (Wi-Fi), the extended Personal Area Network (PAN) ID (Thread) or the network interface instance name (Ethernet) [30, Chapter 11.8.6.3]. From a software perspective, there is no intentional way to read-access network credentials once a Wi-Fi network was added or updated [30, Chapter 11.8.8.4], which prevents trivial exposure of network credentials. The Wi-Fi security type, cipher and credential format is contextually auto-selected during network connection establishment, based on beacons and probe responses [30, Chapter 11.8.8.4]. Nodes may connect to the *last known good* network to skip probing [30, Chapter 11.8.9.1]. A network configuration is stored persistently once the device was successfully commissioned over this network [30, Chapter 11.8.9.2].

4.2.10 MULTI-ADMIN AND MULTI-FABRIC

Multiple administrators of a single node have equivalent privileges with respect to the node [30, Chapter 5.5]. The specification is lacking precise descriptions of how administrators interact with each other, and therefore we assume, based on the description of the operational PKI (see Section 3.5.2), that either of the two multi-admin designs is used: (1) multiple administrators can commission nodes to fabrics entirely independent from each other, and that fabric-related configurations (e.g., ACL) are separate, but it is unclear how the device behaves for device-specific configurations (e.g., energy saving), or (2) RCAs can issue additional ICACs to other nodes to promote them to administrator, so that both RCAC and ICAC administrators share a fabric, and each of them may commission nodes, but only per-node configured administrators may equally administer a node, the RCAC-administrator can revoke ICACs previously issued by itself; in this case having only one device-specific configuration is clearly intended behavior. In (1), a maliciously commissioned device can be commissioned to an attacker-specific fabric which is distinct from the genuine fabric, and in (2), the attacker must compromise or add a malicious ICA, and is not able to influence fabric-specific configurations unless the RCA is compromised, which would compromise the security of the fabric. We assume, that multi-admin design (2) is used due to the considerations made in Section 4.2.11, but could not clearly confirm based on the specification.

An already commissioned node can only join another fabric, if consent by an existing administrator is granted either via Enhanced Commissioning Method (ECM) or Basic Commissioning Method (BCM) [30, Chapter 12.1.2] [30, Chapter 5.6]. In the following, we describe the two administrator assisted commissioning flows ECM, which is mandatory, and BCM, which is optional [30, Chapter 5.6]. The new administrator is required to be in the network for both methods. During ECM, a current administrator assists a node in commissioning with a new administrator. For this, the current administrator generates a new passcode, from which it derives the corresponding PAKE passcode verifiers, which it then distributes to the node. The temporary passcode is not transmitted, but instead outputted visually or auditive. [30, Chapter 5.6.3]. Using ECM, it is possible to commissioning multiple nodes to a new fabric and administrator at once, using the same dynamic passcode, which makes ECM very convenient to use. BCM is optional and similar to the initial commissioning [30, Chapter 5.6.2], and the specification recommends to only support it for devices for which the passcode can be adequately secured [30, Chapter 13.6.2]. Here, the administrator instructs a node to advertise its presence over DNS-SD to allow commissioning over IP. The remainder of the commissioning flow is similar to the initial commissioning of a factory-new device, including the use of the default passcode, but is performed over the IP network, e.g., the operational Wi-Fi.

4.2.11 SECURITY AND LIMITATIONS OF IDENTIFIERS

Matter defines various Identifiers (IDs), some of which are briefly described in Section 3.3.3. Fabric IDs are used to distinguish fabrics and are required to unique within any root CA. The specification foresees a collaboration between the RCA and commissioner for this, which is referred to by the term Administrative Domain Manager (ADM), and is explicitly out-of-scope of the specification [30, Chapter 2.4]. Achieving unique fabric IDs is non-trivial due to the multi-admin, multi-fabric feature. It has not been clearly stated in the specification, but we believe that uniqueness can only ever be guaranteed, if one node may have at most one RCA, because if a node had two RCAs, their ICAs would not be mutually aware of the respective other RCA. Therefore, we assume for the following, that one node may have at most one RCA. Compressed fabric identifiers are used during operational discovery [30, Chapter 2.5.1]. Because compression reduces the entropy of fabric IDs, the mechanism risks collision. Similarly, ADM ensures unique and stable node IDs [30, Chapter 2.4]. In later sections of the specification, their uniqueness is assumed to be guaranteed [30, Chapter 2.5.5.1]. The definition of ADM is out-of-scope. Matter reserves various IDs such as the primordial fabric ID [30, Chapter 2.5.1], an unspecified node ID [30, Chapter 2.5.5.6], the VID 0x0000 for special purposes and `0xFFF{1..4}` for testing purposes as test vendor IDs [30, Chapter 2.5.2]. The PID 0x0000 is used for anonymous discovery announcements, OTA updates which are available for multiple products, and for multi-node commissioning during ECM. Session IDs are used to identify the encryption key out of a key set for a session or group and the encryption or message integrity algorithm used to secure a message [30, Chapter 4.4.1.3]. Their use is described in later sections of this chapter.

4.2.12 SECURITY OF SLEEPY END DEVICES

Sleepy End Devices (SEDs) in Matter follow the same security procedures as regular nodes. They have an *active* and *idle* mode which they can switch between, depending on their involvement in active exchanges [30, Chapter 2.9]. The sleep and active interval of a node are transmitted to the communication party during commissionable, operational and commissioner discovery [30, Chapter 4.3.4] and during session establishments. SED parameters are part of the PASE context, and transmitted in the `Sigma1` and `Sigma2` message of the CASE [30, Chapters 4.13.1.2, 4.13.2.3].

4.2.13 CERTIFICATE AUTHENTICATED SESSION ESTABLISHMENTS

As briefly described in Section 3.3, Matter uses a Certificate Authenticated Session Establishment (CASE) during operational communication. During the CASE, nodes mutually authenticate and generate keys to secure the session's subsequent communication and exchange operational parameters for it, such as the session ID and MRP parameters [30, Chapters 4.13.2, 4.13.2.1]. The CASE implements an SIGMA protocol exchange [50] during which ephemeral ECC public keys are exchanged to generate a shared secret using their own private keys and the retrieved public key. After this, both parties exchange their NOCs to prove their identity, and lastly, sign the ephemeral keys and the NOC to prove they are in possession of their respective NOC private key. Destination IDs are used by the initiator of a session to determine the fabric id and trusted root to select for validation of operational certificates, and which node ID is targeted, and are privacy-protected using IPK epoch keys. An initiator must have knowledge of one the responders IPK epoch keys and a full identity in order to initiate the CASE. Epoch keys are used to exclude a participant from communication by not delivering epoch keys for subsequent epochs, which usually happens when a participant's certificate was revoked. Ownership of an epoch key implicates that the participant was a genuine group member at distribution time [30, Chapter 4.15]. The CASE offers a resumption mechanism to reduce computation and messages required to bootstrap a new session, by omitting signing and signature verification. To initiate session resumption, both parties must know the previous session context: the `SharedSecret` and `ResumptionID` of the previous CASE, and a valid `destinationId` [30, Chapter 4.13.2.2: Generate and Send Sigma1].

4.2.14 MESSAGE COUNTERS

The message counter uniquely identifies a message in the context of security and duplicate detection and is used as message acknowledgement, nonce and for replay prevention [30, Chapter 4.1.1]. Control and data messages use separate message counter domains. Matter implements rekeying to prevent nonce reuse [30, Chapter 4.5]. To prevent nonce reuse after a reboot, message counters of encrypted messages are persisted in durable storage [30, Chapter 4.5.1.3]. The Message Counter is initialized in the range from 1 to 2^{28} to increase the difficulty of traffic analysis attacks, so that the eavesdropper does not know the minimum number of messages that has been sent if the counter is still within the initial range. Rekeying occurs when the message counter wraps, that is, when the

maximum message counter is reached and the value necessarily is required to be lowered. On this event a node discards and re-establishes the session, by which all keys associated with the previous message counter are invalidated and rotated [30, Chapter 4.5.1.1]. Bootstrapping of message counters is performed via unicast UDP [30, Chapter 4.16] either with a trust-first policy, where the initial message is trusted and which is mandatory for control messages and does not use Message Counter Synchronization Protocol (MCSP) [30, Chapter 4.16.3], making it susceptible to replay attacks after reboot [30, Chapter 4.16.1.1], or with a cache-and-sync policy, which is optional and uses a message counter synchronization exchange, protecting against replay attacks [30, Chapter 4.16.1.2]. For control messages that use the trust-first policy, the first authenticated message counter is trusted, which the attacker can exploit by forcing the device to remove the session context and inject an older message, which may occur on device reboot, at which session context may be deleted or recycled [30, Chapter 4.14.1].

4.2.15 SECURE CHANNEL PROTOCOL

The secure channel protocol defines the control plane for secure channel communication and security and is composed of sub-protocols, including Message Counter Synchronization Protocol (MCSP), MRP, PASE and CASE. Security-related status report messages use the protocol identifier of the secure channel and a protocol code for the occurred event and indicate successful and erroneous session establishments, among others [30, Chapters 4.10, 4.10.1, 4.10.3]. There are four unsecured status report messages: `SESSION_ESTABLISHMENT_SUCCESS`, `NO_SHARED_TRUST-ROOTS`, `INVALID_PARAMETER` and `BUSY` [30, Chapters 4.10.1.3]. Nodes can respond with `BUSY` to `Sigma1` or `PBKDFParamRequest` messages that initialize the PASE and CASE, respectively. On receipt, the initiator waits for the minimum time specified in the `ProtocolData` of the `BUSY` status report, but no more than a minute, then generates a new request with new randomized values [30, Chapters 4.10.1.5].

4.2.16 GROUPCAST COMMUNICATION

Matter offers multicast communication via groupcast sessions, in which all participants share the same symmetric encryption key. The groupcast session context consists of the fabric index, Group Identifier (GID), source node ID, source IP address, source port, operational group key and group session ID, of which source IP address and port can be used for unicast responses. There is a risk of undefined behavior when a node views or participants in two or more distinct groups that share the same group identifier. Because of this, Matter recommends that clients ensure uniqueness of GIDs [30, Chapter 9.11]. Operational group keys serve the purpose of authenticating peers, integrity and encryption, and are managed by the group administrator, who is responsible for ensuring that nodes are grouped together appropriately with respect to their applications and security requirements, and are used in epochs [30, Chapter 4.15]. For each epoch, one operational group key is derived in a KDF using an epoch key as input key, the compressed fabric ID as salt. At least one, but at most three epoch keys are kept in the group state [30, Chapter 4.15.3.2]. The epoch keys are generated using a DRBG [30, Chapter 4.15.3] and used to enforce group membership. They are

managed in the *group key management cluster* which is scoped to a node and managed by the group administrator [30, Chapter 11.2.1]. The commands used to update or revoke epoch keys are called by the group key administrator and is idempotent [30, Chapter 4.15.3.2]. Epoch keys have a start time but no end time, and the current epoch key is the key with the latest start time that is not in the future. Epoch-key derived operational group keys are accepted by nodes as long as they are in the group state, regardless of any start or end time [30, Chapter 4.15.3.1]. The key rotation rate is expected to be in the order of days or weeks, but the decision is passed on to the manufacturer [30, Chapter 4.15.3.3]. Matter mitigates DoS attacks based on key distribution jamming by instructing nodes that are not capable of maintaining a synchronized clock to use the epoch key with the second newest start time, and by overlapping epoch keys. Furthermore, nodes are not required to be in possession of active epoch keys to be able to receive new keys, since they are updated by the key distribution administrator via unicast [30, Chapter 4.15.4]. Multiple operational groups may share the same operational group key and can be used to create logical subgroups over a shared security domain [30, Chapter 4.15.1].

4.2.17 CERTIFICATES AND KEY MATERIAL

As briefly described in Section 3.5.2, the device manufacturer issues a unique DAC to each device. During the DA procedure, the commissioner generates a random 32 byte attestation nonce using a DRBG and sends it to the commissionee before requesting the attestation information, which are returned by the commissionee with a signature [30, Chapters 6.2.1, 6.2.2.1]. The commissioner should validate the attestation information by checking, among others, that the start time lies in the past and that neither of PAA, PAI and DAC are revoked [30, Chapter 6.2.3].

Firmware information is used to communicate that a device implements a secure subsystem protecting the DA private key, allowing to validate the firmware on boot, preventing a device from booting modified firmware. It is an optional component of the DA information [30, Chapter 6.3.2]. Commissioners can use the firmware information to determine whether the firmware is allowed to be executed on the device, to check for revocation and known vulnerabilities. If the commissioner or administrator detects a corrupted firmware version, it may refuse commissioning and optionally inform the user [30, Chapter 6.3.2].

4.2.18 ACCESS CONTROL

Access control ensures that only authorized nodes are permitted access to certain application-layer functionality. For this, the administrator creates and maintains a consistent and distributed configuration of an Access Control List for each node in the fabric. The access control is rule-based and denies access by default, and is empty after factory reset [30, Chapters 6.6.2, 6.6.3]. Therefore, access is denied unless the required privilege level is explicitly granted. During commissioning, the administrative access privilege is implicitly granted to the administrative subject [30, Chapter 6.6.2]. Privilege levels are *view, proxy view, operate, manage* and *administer* and are applied to subjects, which are either a passcode, passcode id, source nodes or destination groups [30, Chapter 6.6.2]. Wildcards are empty subject lists of access control entries and grant access to any subject.

There is a risk that manufacturers implement a removal of subjects one by one such that it result in a wildcard when the last subject is removed [30, Chapter 6.6.2.2]. Manufacturers are allowed to extend the access control, but it is expected that they improve the overall security [30, Chapter 6.6.2.6]. Changes of the ACL are reported as event records [30, Chapter 9.10.7.1]. which allows to deploy network-based Intrusion Detection System (IDS) to identify irregular ACL changes. If the administrator cannot reach a node when altering an ACL entry for a group, it is possible that the node grants obsolete privileges until it received the update. Administrators can revoke privileges of other administrators [30, Chapters 6.6.3, 6.6.4]. For bridged devices, the administrator cannot distinguish between individual devices, but will instead grant the privilege to the bridge, which inherits the privileges to the bridged devices [30, Chapter 13.6.8].

4.2.19 DATA AND INTERACTION MODEL

Matter's data model is based on Zigbee's data model architecture (revision 8) [30, Chapter 7.1.3], and its interaction model is based upon the Zigbee cluster library [30, Chapter 8.1.3]. Data model elements have access qualities defining how the field can be interacted with, such as read, write, fabric or privilege-based access, and timed interactions.The duration during which an action message is valid is limited by timed interactions, which prevent replay attacks [17] [30, Chapter 7.6.10]. During timed interactions, interactions are only valid during a specified interval. Timed interactions are achieved by a 2-phase commit in which the target of the interaction sends an authenticated *Timed Request Action*, defining a timeout interval up to 2^{16} milliseconds (at most roughly a minute). The receiver replies with a status response `SUCCESS`, on which the interval is started [30, Chapter 8.7.4].

The data model describes the persistence across restarts, reboots and upgrades. Persistent data is reset to its original state upon factory reset [30, Chapter 7.12.1]. In general, all Matter devices and nodes must support factory reset used to remove any security- and privacy-related data created during or after commissioning, apart from data that is explicitly required to persist across resets [30, Chapter 13.4, CM15].

Events are implicitly readable records similar to log entries, that reflect any changes of a node to an observer [30, Chapter 7.14]. All event records have an node-scoped, persistent and monotonically increasing event number, a timestamp either in the system time or epoch time in microseconds and a priority. Event records are buffered on the node to which they refer, and in case of insufficient space, they overwrite events within the same priority level, but no other events [30, Chapter 7.14.3]. Event records are prioritized as debug, info or critical, the latter of which is reserved for events that impact physical safety or reliable operation.

4.2.20 BRIDGES

Bridges connect Non-Matter devices to a Matter fabric, allowing them to interact with Matter devices and be controlled by them [30, Chapter 9.12.1]. The certification of bridges is done per device types they support. For some applications, bridges are able to emulate features that are common for

17 The Matter specification says 'timing attacks', which we believe to be a typographic error.

Matter devices of the device type to support, which the bridged device does not support by itself, such as lighting patterns [30, Chapter 9.12.11]. Bridged devices are uncommissioned, do not obtain cryptographic material on their own and have no ACL in the context of Matter. They are in full control of the bridge, which adds and removes them from the network [30, Chapter 9.12.8]. The bridge has a single access control cluster for the entire node, that contains the access control entries for the bridged devices, each of which is bound to exactly one distinct endpoint of the bridge [18] [30, Chapter 9.12.9].

4.2.21 SUBSCRIPTIONS AND PROXIES

Matter offers a publish-subscribe model in which a subscriber receives updates from the publisher in form of change notifications, such as events [30, Chapter 8.5]. Subscriptions are ephemeral 'sessions' [19] and are removed if the device is factory reset, rebooted, if network connectivity issues occur or when resources need to be reclaimed [30, Chapter 9.3.1.1]. The maximum number of subscribers can be very limited, for example it is possible that a low-power device offers at most one subscription. Using proxies, the maximum number of possible subscribers can be increased by multiplexing multiple subscriptions into one [30, Chapter 9.15]. It is recommended for clients to select the proxy with the least hops [30, Chapter 9.15.7.5]. Non-proxy subscriptions are evicted first to make space, if a source is not able to handle another subscription. The evicted clients then undergo reassignment, during which they are likely assigned to a proxy instead of the source. Proxies subscribe to other proxies if possible, else directly to the source [30, Chapter 9.15.7.6]. Proxies are certified by the CSA to ensure consistent behavior, though certification is not mandatory [30, Chapter 9.15.9]. The specification recommends to commission proxies into all fabrics in a multi-fabric setting [30, Chapter 9.15.8.3]. To prevent malicious and unattested device from acting as proxies, admins can specify valid and attested proxies to a source as a means of authentication [30, Chapter 9.15.10.1]. The proxy discovery commands is sent over multicast and encrypted with a specific group key [30, Chapter 9.15.10.2]. Administrators can enable proxies by enabling the *proxy view* privilege. The specification recommends to grant the privilege for the entire source node instead of single clusters [30, Chapter 9.15.10.3]. The subscription system auto-balances itself [30, Chapter 9.15.7.9], and in a single-fabric setting, only one proxy should directly subscribe to the source [30, Chapter 9.15.8.2].

4.2.22 GENERAL DIAGNOSTICS CLUSTER

The general diagnostics cluster provides standardized diagnostic metrics to assist in diagnosing problems. It can reports various hardware faults such as sensor, ratio or network faults and temper detection, e.g., it can report network faults caused by jamming or connection establishment

18 As a clarification, access *to* bridged devices is granted individually per device, but access *from* bridged devices is done per bridge.

19 To our confusion, the single quotes are also used in the specification, but it is not described in how far subscriptions differ from other sessions.

failures [30, Chapter 11.11.11]. The software diagnostics cluster also provides standardized diagnostic metrics to assist in diagnosing (potential) problems, and provides information such as watermarks [20], stack, heap [30, Chapter 11.12]. Matter offers distinct diagnostics clusters for Thread [30, Chapter 11.13], Wi-Fi [30, Chapter 11.14] and Ethernet [30, Chapter 11.15], offering standardized diagnostic metrics that are specific for the underlying communication channel. Examples are the Thread routing table, the Wi-Fi security type or collision counts.

4.2.23 OVER-THE-AIR DEVICE-FIRMWARE-UPDATES

The specification recommends to check for updates on a daily basis due to security concerns [30, Chapter 11.19.3.2]. Before serving an update, OTA providers should obtain (informed) user consent [30, Chapter 11.19.3.4]. The update process can be discontinued, which can be used to prevent damage in case of an update that has significant negative impact on device functionality [30, Chapter 11.19.3.6]. Over-the-air Device Firmware Updates (OTA DFUs) are mandatory [30, Chapter 13.5] for manufacturers to implement and can be obtained and installed by OTA requestors, who obtain the updates from OTA providers. OTA providers must support the Bulk Data Exchange (BDX) protocol to allow OTA requestors to download the image without access to the public internet [30, Chapter 11.19.2], but vendor-specific legacy out-of-band methods are explicitly allowed [30, Chapter 11.19.1]. OTA providers can be discovered using runtime service discovery, which is out-of-scope of the specification [30, Chapter 11.19.3.1]. When a (native) update is available, the requesting node is provided a URI that points to the downloadable image.

The header of the software image file contains general information such as the VID and PID to which the update applies, a payload size, the image digest and an URL for release notes [30, Chapter 11.20.2]. OTA providers may cache updates, and serve them from cache if they have the correct checksum, checksum algorithm and file size [30, Chapter 11.19.3.3]. The certificate used to sign the update can be pre-installed on the requestor, or received over-the-air, which is out-of-cope of the specification and vendor-specific [30, Chapter 11.19.4.2].

The Bulk Data Exchange (BDX) protocol used for non-legacy updates is carried over any reliable supported Matter transport such as BTP, TCP or MRP and must be secured over either a PASE or CASE [30, Chapter 11.21.4]. The BDX version is decided by the responder [30, Chapter 11.21.7] based on the highest compatible version of the initiator. The responder in this context is the OTA requestor, who initiates the update request, but the contained BDX exchange is initiated by the OTA provider [30, Chapter 11.19.3].

OTA DFUs are provided independent of their compliance status, which OTA requestors are not required to display to the user and are allowed to ignore [30, Chapter 11.19.3.3]. Matter allows roll-out of non-standard software images [30, Chapter 11.19.5.2].

20 Watermarks in terms of memory consumption. To the best of our knowledge, Intellectual Property (IP) watermarks are not supported by Matter.

4.2.24 IN-FIELD UPGRADE TO MATTER

The manufacturer can provide new firmware to the device, together with an associated Certification Declaration and a unique Device Attestation Certificate to in-field upgrade a Non-Matter device to a Matter device. The device is restarted to enable the new firmware, turning it into an uncommissioned device [30, Chapter 5.8].

4.2.25 SECURITY MEASURES OF DEVICE MANUFACTURERS AND RECOMMENDED COUNTERMEASURES

The Matter specification describes various security recommendations [30, Chapter 13.6] and countermeasures to threats [30, Chapter 13.7], which are optional to implement.

We noticed that some of the countermeasures are described so vaguely that they are similar to tautologies. For example, for T60/CM62 [21] the specification advises to use protection against physical attacks to protect against physical tampering, while for T117/CM113, incorrect fusing is countered with correct fusing, and T153/CM36 prevents adding untrusted root certificates to a node, by ensuring that the commissioner only adds trusted root certificates to a node. We suggest that future revisions of the specification ensure to enlist meaningful countermeasures for all listed threats.

We found several countermeasures that do not adequately protect against the particular threat. In the following, we briefly comment on some of them, before progressing to the symbolic verification and discussion. There is a threat that a device might advertise information that the attacker can use to identify vulnerabilities or reconnaissance of a facility or user, which can be mitigated by data minimization, user control of unsecured actions, deletion of orphaned state information and session establishment time-outs [30, Chapter 13.7, T7]. We argue that session establishment timeouts only limit the time during which the information is accessible, while still providing the same information, meaning that the time window during which the attack is possible is reduced. Of the mentioned countermeasures, only data minimization can protect against reconnaissance by persistent threats.

The administrator should respond to reports of attacks and malfunctions by deny-listing, user notification, changing of group keys and channel hopping [30, Chapter 13.7]. It is not described, however, how the administrator is supposed to detect attacks, which is not trivial in most cases. Furthermore, if not implemented properly, the attacker can abuse the deny-listing mechanism to entice the administrator to perform a DoS attack on the communication of genuine nodes by sender spoofing, which is possible for any message that is not CASE-secured.

Devices that are marked for destruction might be reused in a network, which can be mitigated by revoking their credentials and privileges [30, Chapter 13.7]. It is unclear whether device credentials mean the DAC or NOC. In these situations, it would be favorable to revoke the DAC and factory reset the device, to ensure that any user data and network configurations are removed, and to ensure that the device is permanently prevented from joining Matter networks that require DA verification.

21 T60/CM62 is read as 'Threat 60, with corresponding Countermeasure 62'

4.3 SYMBOLIC VERIFICATION

By encoding the syntax and semantics of security protocols in formal language, symbolic models, also referred to as Dolev-Yao models, can be used to model protocol execution and an attacker with full control of the network. Security properties are encoded as reachability or equivalence. The general problem of verification is undecidable which might lead to non-termination for specific protocols or properties, but symbolic verification contributes towards solving the problem of asymmetric workload of the defender and attacker, where the defender has to defend against all attacks, but the attacker has to find at least one attack. The absence of an attack can be defined as a lemma, which, if there is no solution, is proof, and else, the attack exists. However, the symbolic model execution can also run out of time or memory, which might require a security researcher to interactively model the protocol flow. In the symbolic model, cryptographic primitives such as RNGs, encryption, signatures or hash functions are assumed to be perfect, which can be seen as a major limitation of the model when assessing the security of a protocol. Since Matter is built with crypto-flexibility in mind, it can be argued that Matter is secure if the Dolev-Yao model is secure and if for each cryptographic primitive used by Matter a variant exists that is secure and are either already used or are used in the future when its preceding cryptographic primitive needs replacement due to security concerns.

To assess the security of Matter, we have translated plain SPAKE2, Matter's version of SPAKE2+ used during the PASE and the SIGMA exchange of the CASE into symbolic models for the Tamarin prover. The PASE and CASE are the two session establishments used in Matter to derive shared secrets, and it can be assumed that Matter would security-wise be broken, if any of the two protocols failed to meet the security requirements. Under the assumption that the translation is correct, it is shown that the exchanges work. Any security lemmata did not terminate and therefore it was not possible to prove that the protocols are secure. The Tamarin theories used are available for future research in Appendix D.

5 DISCUSSION

> *Despite continued security problems, the IoT will spread and people will become increasingly dependent on it. The cost of breaches will be viewed like the toll taken by car crashes, which have not persuaded very many people not to drive.*
>
> — RICHARD ADLER

During our analysis of Matter's specification, we did not find any direct vulnerabilities. All in all, Matter implements great security. In the following, we will discuss various aspects which we found to stand out the most, elaborate an deviations and implementation-specific weaknesses that can lower the security of Matter devices and networks, and provide mitigations and improvement suggestions. Cases in which the administrator device is compromised are not considered unless explicitly stated otherwise. In general, a compromise of the administrator compromises the security of the entire fabric, because the administrator issues certificates and identifiers, configures the devices and ACLs, is in possession of network credentials and distributes group keys. We start with a description of cryptographic primitives and continue from top to bottom through the remainder of the specification. We refrained from rating the likeliness of a threat unless stated otherwise, and instead focused on exploring the set of specification-focused attack options. The entire discussion is based on a theoretic analysis of the specification, and the weaknesses described have not yet been confirmed in practice, which is left for future work.

5.1 REMARKABLY STRONG CRYPTOGRAPHY

Matter uses remarkably strong cryptography as described in Section 4.2.1, which we discuss in the following subsections. Note that not all devices have to support all cryptographic primitives, if a precomputed value can be stored on device [30, Chapter 3].

5.1.1 RECOMMENDATION OF ADDITIONAL ACCEPTANCE CRITERIA FOR RNGS

Matter uses and recommends secure RNGs in form of a TRNG-seeded DRBG. For improvement, we recommend to add more acceptance criteria for RNGs, such as statistical properties, randomness, type of the entropy source (analog or digital), stochastic model, configuration and potential erratic results as well as a proof of quality as minimal and mandatory requirements. Furthermore, Matter could restrict RNG implementations to quality-proven TRNGs and DRBGs. However, it can be argued that split-testing might speed up the evolution of secure algorithms, and therefore, freedom within reasonable boundaries might be the preferred strategy, a contradiction of security-best practice and innovation which we hereby acknowledge. In any case do we deem it necessary to excessively test the RNGs used by devices under certification, because most of Matter's security is based on them. This includes observing the behavior after reboots, crashes, and other events

that could reduce entropy. Due to the higher costs of good RNGs, we are expecting manufacturers to short-cutting security of RNGs, which could be partially mitigated when manufactures can integrate Matter-certified SoCs that include suitable RNGs.

5.1.2 HASHES AND HMACS ARE SECURE

We consider the use of hashes in Matter to be secure as discussed in Section 4.2.2, even though the SHA-256 algorithm used is vulnerable against length extension attacks in general. We recommend the developers to keep this in mind if future adjustments to Matter's cryptography are made. The use of HMACs in Matter is secure, even though there is one scenario, where the salt exceeds the maximum size of 512 bits: during the calculation of the S2K in the CASE. But since all HMACs are used to derive shared secrets, any attack would result in protocol abortion, similar as any attempts of length extension attacks on hashes do.

5.1.3 MATTER COULD LEVERAGE THE FULL ADVANTAGES OF AEAD

Matter uses AEAD to secure confidentiality, integrity and privacy of messages. We showed that AEAD is used securely by arguing why neither bit-flipping attacks nor reused key attacks are likely to succeed, and by showing that any attacker being able to eavesdrop on the SIGMA exchange during the CASE can reduce privacy, but not security. Furthermore, we showed why a breach of privacy encryption does not weaken the security of message encryption, and that Matter is secure against modification of any fields sent in plaintext or privacy-encrypted. However, we did not exclude the risk of chosen-IV attacks and side channel attacks, for which it is left for future research to determine potential attack scenarios and impact of the attack. We observed that Matter uses a single function for decryption and verification of incoming ciphertext. While this is no vulnerability on its own, we recommend to add requirements on how and where processed plaintext or ciphertext originating from receive buffers is stored during the process, to leverage the full advantages of AEAD. We noticed that the privacy encryption used in Matter should follow a recommendation for AES-CCM although the encryption uses AES-CTR, but left further analysis of the issue to future research.

5.1.4 EXCEPTIONALLY STRONG PAKE, MIX-UP OF VERSIONS

Matter's use of SPAKE2+ as the PAKE is embedded securely into the commissioning flow and reduces the risk to targeted attacks due to the high amount of resources and time required to perform the attack. We discussed the remaining risk of offline precomputation attacks and how Matter aims to prevent them, and came to the conclusion that the risk is negligible unless the target implements a significantly flawed RNG. In general, offline precomputation attacks are a major drawback of PAKE protocols like SPAKE2+. They are interesting for the attacker in those cases where the commissionee's setup code, or passcode, is unknown to the adversary. Manufacturers could easily add additional difficulty to the attack by randomizing salt and number of iterations used for the PBKDF

on an individual device level. Furthermore, there is a risk that manufacturers chose passcodes statically or predictable or with lower entropy, which might not be detected during certification. Dynamic passcodes could prevent this, but introduce other attack surfaces such as side channel attacks and therefore we do not advise against the one or other.

SPAKE2+ was proven secure in a preprint by Shoup [79], one of the original co-authors of the SPAKE2+ paper [15]. The analysis was performed on a modified version of SPAKE2+ in a server-client scenario [1]. Shoup claims that SPAKE2+ is secure against a compromised server (commissionee) for which the password file (onboarding payload or PAKE values) was leaked. The goal of the security analysis was to show that the adversary cannot log into the server. It is not safe to assume that Shoup's security analysis is applicable to the use of SPAKE2+ because it was performed on a variation of SPAKE2+ that Shoup himself described similar to, but not identical to the specification of Taubert and Wood [82] which is referred to by Matter. On the one hand, Shoup claims the variation to be a generalization of SPAKE2+, which supports the assumption of applicability. On the other hand, since Shoup is one of the co-authors of the original SPAKE2+ paper, we have to assume bias and must conclude that there is not yet any independent academically conducted security analysis of SPAKE2+. Furthermore, the proof was published in a preprint and was therefore yet neither successfully peer-reviewed nor accepted by a journal. This does not ultimately reduce its validity, but rather implies that a grain of skepticism is appropriate. Still, it is not safe to assume that security analyses of generalized key exchange algorithms can be inherited by their implementations, unless it was shown that the implementation was perfectly following the protocol and when not considering any side channel attacks and similar forms of bypassing mathematical assumptions. Regardless of perfect applicability, we present our core take-aways from Shoup's work: Shoup claimed that the adversary would certainly be able to impersonate the server to a client. For Matter, this means that a commissioner is not able to detect fraudulent commissioners using SPAKE2+ alone. We note that there is a significant difference in context when comparing a typical server-client model against the key exchange in IoT, because in Matter, it is not predetermined who the communicating parties are, i.e., while a user logging in to a server usually knows who the server is, a commissionee is agnostic about its communication parties and would accept any commissioner, unless a manufacturer explicitly restricted two devices to exclusively be in use with each other. From a user perspective, the commissioner can be impersonated against a commissionee, because there is no mutual authentication during the PASE and generally no means to distinguish a malicious device from a genuine device and no mechanism on the commissionee's side to prevent commissioning to a device that is in possession of the commissionee's valid onboarding payload. Shoup [79] claims in his preprint that an offline dictionary attack cannot be initiated before the password file (onboarding credentials or PAKE values) is leaked, unless the attack is directed at a particular client-server pair. Such an event would negate the benefits of using a PAKE protocol instead of traditional password-based protocols layered on top of a one-sided authenticated key exchange. However, Taubert and Wood warn in their specification draft against Unknown Key Share (UKS) [11] [84] attacks if the exchange was performed with not implicit identities [83].

SPAKE2+ is used without identity information in the Context, from which the shared secret is derived. In other words: The shared secret is bound to a context. We assume that in Matter,

1 In the Matter SPAKE2+ context, the *commissioner* is also referred to as prover, initiator or client, and the *commissionee* is also referred to as verifier, responder or server, depending on document, author and section.

the identities are not implicit: The term 'implicit identities' is ambiguous and, to the best of our knowledge, not well-defined. It might apply to a situation where public keys or fingerprints are used to identify. Or it could describe a situation, where there is a direct link between two parties, and the link itself proves the identity of the other party because only exactly this other party could have used the link, which might be a valid assumption under another assumption that only the legitimate commissioner possesses the onboarding payload. Depending on whether we assume the BLE or Wi-Fi channel used to perform the PASE to be secure or not, and whether we assume that a link over BLE implies identities or not, we end up with either implicit or non-implicit identities. We abbreviated the consideration by sticking with the Dolev-Yao attacker, or attacker within the network, because Matter claims to be secure independent of the communication technologies it operates on, and because commissioning is possible over a (potentially public) Wi-Fi network, and instead focused on the essence of the consideration [30, Chapter 4.3.1].

We noticed that Matter mixes draft versions 2 and 8 of the SPAKE2+ draft, resulting in Matter using confirmation keys that are 64 bit long instead of 128 bits used in the recent SPAKE2+ draft, an issue which might require further investigation. Furthermore, we believe that a residual, but negligible risk of MitM and UKS attack exists, but argue, that since the PASE bootstraps security in the operational network, and because of the security-privacy trade-off, this is hardly fully preventable, because in the end, two pairing devices must start to trust at some point. In Matter's case, the initial point of trust of the commissioner for the commissionee is the commissioner's knowledge of the passcode. Even though Matter already provides a very strong set of security, we therefore advise CSA to take further precautions and audit the passcode generation process such that the certification procedure can guarantee that manufacturers do not use predictable or static passcodes or passcodes with significantly less entropy, because this would greatly reduce the security of Matter. Furthermore, we recommend that manufacturers take further precautions by randomizing salt and number of iterations used by the PBKDF on an individual device level to maximize the security precautions during the PAKE by reasonably cheap measures. In summary, we come to the conclusion that Matter implements remarkably strong cryptography, which depends on the secrecy of the passcode (or derived PAKE values) and a strong RNG. To the best of our knowledge, we regard Matter to be secure against cryptographic attacks and do not expect this to change the next centuries, due to the high entropy [2] of the randoms and keys used in Matter.

5.2 MUCH OF MATTER'S SECURITY DEPENDS ON PASSCODE SECRECY

We described the device passcodes in Section 4.2.4 and showed that the passcode entropy is incorrectly specified as 27 bits, although it approximates 26.575 bits, due to a confusion between 'key length' and 'entropy'. We do not expect the reduced entropy to degrade security, because Matter has designed key derivation such that each party contributes a random of which each by its own provides sufficient entropy, even if the entropy of all other factors was removed (see Section 4.2.2). Still, it would be best if CSA confirmed that the reduction of entropy does not weaken any of their security assumptions, and to correct the imprecision in the specification, because the confusion might

2 Note that we did not consider quantum-safety, nor did we consider any unexpected innovations that significantly increase computational power more than is predicted by Moore's Law.

cause manufacturers to make erroneous assumptions and follow-up decisions on passcode security policies. We observed that anyone who is on possession of the passcode can eavesdrop on the device's commissioning procedure or perform a MitM attack on it, and control and configure the device unless manufacturer-specific precautions are taken, including the provisioning with compromised network credentials. From our personal experience, home IoT is usually not physically access-restricted and monitored, and therefore we recommend to store the onboarding payload separate from the device (in packaging), and watermark it such that most attackers are prevented from faking it and to prevent an attacker from swapping the passcode, which can be used to trick a user into commissioning an attacker-controlled device. By sealing the in-packaging onboarding payload, an attacker can be prevented from obtaining the code prior to device sale without the user noticing. Both dynamic and ephemeral passcodes are protected against passcode leakage, by allowing the user to offline-generate a dynamic passcode, or request an online-generated ephemeral passcode during the custom commissioning flow, which the attacker cannot know beforehand. It is unclear whether Matter Version 1.0 already offers dynamic passcodes aside from their use during ECM. It is advisable that the Matter specification requires manufacturers to sufficiently educate their users on how to securely store the payload. We noticed that only dynamic and ephemeral passcodes offer non-reduced security when being resold, in comparison to static passcodes, for which the new buyer usually does not know who is in possession of the passcode. We hope that Matter aims to further close the gap between the desire for security and sustainability. We recommend to add examples of how read-out of the onboarding payload in-packaging can be prevented if NFC tags are used, e.g., RFID NFC signal blockers or proximity-based approaches, and the certification procedure should ensure that a read-out from outside of the packaging is not possible. Sealed signal blockers are more favorable, though, because they can prevent read-out from suppliers, such as a salesclerk, and provide protection even when packaging is opened in store. We therefore suggest that the certification procedure assesses how the passcode, onboarding payload and PAKE verifiers are secured during manufacturing, shipping and retail, and provide best practice examples as well as mandatory requirements on their security. The certification procedure should also ensure not to certify any devices that allow NFC tag read out from closed packaging or swapping of NFC tags, and additionally should audit the passcode generation function to ensure sufficient randomness and to prevent the generation of forbidden passcodes. Furthermore, we recommend to make countermeasure CM89 mandatory, meaning that any on-device passcodes must be designed such that they are not trivially photographable, e.g., by hiding them in the battery compartment, because it provides a good trade-off between usability and security, even though the countermeasure is not suitable to fully prevent attacks on codes that are displayed on the device, as discussed in Section 4.2.7. If a manufacturer chooses to permanently attach the passcode to a device such that it cannot be trivially removed, for example printed on the cover, it might be difficult for the user to prevent access to the code, and there is a risk that users come accustomed to codes being on-device. There is a risk that a demand arises for this practice due to ease of use, which could greatly reduce overall security and privacy, especially when users don't understand the security implications of attached passcodes. A similar habit can be observed with Wi-Fi network credentials, which are commonly stored on routers and often enough remain unchanged by their users. We argue that secure devices should not only protect against dangers from outside the flat, but also against partner and parental abuse, as well as nosy children. We base this recommendation on the claim of Harkin et al. [44], who

found that an often-forgotten incentive of spyware is partner abuse. Therefore, we advise against passcodes that are placed on-device.

We found a flaw in the specification that risks access of arbitrary applications on a multi-function device (e.g., both commissioner and commissionee are two distinct applications on a smartphone) to obtain the passcode, if the recommended copy-and-paste functionality uses the OS clipboard. If this is done, the passcode is not guaranteed to be secure against malicious applications that are installed on the multi-function device, that have permission to access the clipboard. In worst-case, the attacker obtains the ability to control the node for which the passcode was leaked and lock out the legitimate user, and on rooted devices, where the attacker can use a network sniffer [3], they can eavesdrop on commissioning and potentially obtain network credentials. This easily avoidable risk strongly contradicts the CSA's claim to secure the user's security and privacy (see Section 3.6) and thereby we recommend to not allow any unsecured copy-and-paste functionalities such as the clipboard.

In 2013, Fahl et al. [37] described credential stealing attacks on clipboards. We argue, that if this feature was widely used, attackers could be incentivized to mass target Matter devices remotely, because the attacker could create an application that listens to the clipboard and collects any items that might be passcodes, and then act as a commissioner against the target device, if it is in either initial commissioning mode or in BCM. We provide an example for Android: In 2017, Pearce demonstrated the issue by implementing a Proof-of-Concept (PoC) application [43] that targeted Android below version 10. In Android 10, which was released in 2019, access to the clipboard is only possible for applications that have focus [42], which would require the user to open the malicious application after the passcode was pasted, but the risk persists, especially if the malicious application is used intensively. Since version 12, clipboard access notifications are introduced [41] that further mitigate clipboard-based attacks. However, we want to stress out that Matter can relatively easy forbid the use of unsecured copy-and-paste functions by black- or whitelisting specific functions for the most commonly used operating systems, which is reasonably low effort, but good guidance for developers. Obtaining the network credentials allows an attacker to directly connect to a network instead of requiring to persist malware on a genuine device, if the attacker is able to get in proximity. In targeted attacks, this can be considered an elevation of privileges. There is a risk that the credentials are exposed, similar to how wardriving exposes unsecured networks [94]. Using the clipboard would enable the attacker to potentially steal user data and, depending on the device functionality, integration into a botnet (e.g., for a DDoS). Reardon et al. [70] argue, that some applications are already known to use covert channels and side channels to steal user data, while Harkin et al. [44] argue, that the threat of spyware used for either profit or espionage is real due to the commodification of mobile phone surveillance. Furthermore, monitoring a user's IoT data traffic or IoT application data could be the next generation spyware or data mining approach, especially when the IoT market converged to uniformly adopt one protocol with seamless integration to the internet, which reduces development costs and allows scalable solutions for both attacker and defender side, which can be regarded a double-edged sword. Using the clipboard would also open the attack surface to code injection attacks by clipboard data manipulation [98]. We therefore strongly advise that the use of copy-and-paste of passcodes is restricted to secure and unicast methods, such

3 The sniffing itself is not a risk related to Matter.

as data sharing between applications (e.g., via *Content providers* [4]).

Because access to a Matter node is protected by its passcode, we reason which entities may be in possession of the static passcode, depending on the security mechanisms in place to protect it: It can be assumed that each buyer, owner, user, vendor and manufacturer and logistics worker, visitor, observer or room mate is potentially in possession of the passcode [5]. As argued above, for devices that allow to copy-and-paste the passcode, it might potentially be known by any application on the device. Regarding vendors we should distinguish between demonstration models, opened and closed packaging as well as on-device and in-packaging passcodes. Obtaining the passcode by peaking into the packaging can be prevented by sealing the passcode such that the user can demand replacement if the seal is damaged, which also prevents frequently involved third parties such as salesclerks from obtaining the passcode. For demonstration models where the vendor is potentially the first to commission the device, an individually sealed 'demonstration' passcode could be used, which is automatically disabled when the user initiates commissioning with the correct 'main' passcode. However, this would only limit passcode knowledge in scenarios, where a device is sold *once*, but if a device is sold multiple times, its security is still reduced when compared to new devices. For sustainability reasons, multiple iterations of selling should not reduce device security. Dynamic passcodes for initial commissioning can potentially solve the issue.

As already explained, the passcode recovery requires a proof of possession and a rotating device identifier, the latter of which is known to the Dolev-Yao attacker. We found that the process is underspecified such that it has potential to greatly weaken security assumptions. Proof of possession is a hard problem that will most likely cause security issues, especially when service staff is not security-aware and take short-cuts once in a while such that an attacker can obtain sensitive information even when they do not have the proof of possession. It remains an open question, how recovered device passcodes are transmitted to the (alleged) device owner. An attacker might target the transmission of the recovered passcode [6]. Whenever a passcode is recovered, there is a risk that the service staff obtains the passcode, as well. Malicious service staff could target familiars, or burglars might hire as service staff to obtain passcodes of door locks [7]. We therefore recommended that the CSA defines minimal requirements or restrictions of what a legal proof of possession is, and how transmission of both passcode and proof of possession must be minimally secured. Furthermore, we suggest to secure passcode recovery such that no blind trust in service staff is required. The passcode, and therefore the PAKE, is inherently not secure against manufacturer staff unless sophisticated means of securing the process are taken, e.g., by extensive monitoring of the process and active tracking of potential abuse. It is not specified whether a device issues a warning when a passcode is recovered, so that there is a risk that malicious recovery is not noticeable. Therefore, we recommend that future revisions of the specification add minimal requirements on the proof

4 https://developer.android.com/guide/topics/providers/content-providers.html [Last accessed: 2023-01-20]

5 E.g., buyer Alex buys a present for owner Bob, who later lends it to a friend, the user Charlie. A year later, they sell it to Denis, who becomes the new buyer, owner and user. Note that the device vendor (e.g., IKEA) is also a device owner at some point, as is the manufacturer (e.g., Apple).

6 For example, transmission via e-mail that is only transport encrypted, but not application encrypted, potentially allows the involved e-mail providers to obtain the passcode.

7 Similar breaches are known for health data, where workers in healthcare spied on their relatives, acquaintances, coworkers or famous people, a problem that is greatly mitigated by extensive monitoring of access to patient records.

of possession, transmission of recovered passcodes, security against malicious manufacturer and service staff and optionally notifications on passcode recovery.

In summary and despite our suggestions for further improvement, we believe that Matter provides a strong passcode model, by which devices are sufficiently protected, especially when ephemeral passcodes are used. It would be even more favorable, if the user was granted choice between the different passcode models (static, dynamic, ephemeral), because these methods constitute a trade-off between security and privacy. However, it can be argued that user choice is granted via interoperability. We hope that future revisions of Matter are a bit more restrictive on how passcodes are distributed.

5.3 THE USUAL STRUGGLE WITH TIME SYNCHRONIZATION

Matter uses time synchronization and certificates for device authentication, as summarized in Section 4.2.5. If a node believes the current *Last Known Good UTC Time* to be incorrect and has a good time value from a trusted source, it may adjust the Last Known Good UTC Time backwards. It is not explained how and when a device performs the check. As mentioned earlier, when a node switches to synchronized time, it should recheck security material and existing connections [30, Chapter 3.5.6.1]. This is important, because certificates of communication partners may have expired in the meantime. It is thinkable that an attacker could try to force a device to discard its Current Time and fall back to the Last Known Good UTC Time or manipulate either of which, causing a device to trust an expired certificate. If an attacker was able to force a device to use Last Known Good UTC Time, he would be able to circumvent certificate expiration, and if he was able to trigger synchronization, he might be able to cause asymmetric workload at the target's side, which could facilitate a DoS. Furthermore, the attacker might be able to gain control over the device time for a short time, and thereby influence validity checks. We argue that time synchronization cannot perfectly be guaranteed, because NTP-based solutions are potentially vulnerable against connection failures and amplification DDoS attacks on the surrounding network infrastructure. While this is an inherent issue of distributed systems and PKIs for which we do not consider it a Matter-specific weakness, we stress out that manufacturers should include the possibility of devices being unexpectedly not updated in time in their security models. Matter means to prevent a single point of failure by offering redundancy in time synchronization services with multiple powered nodes. All in all, we did not find any unexpected or (easily) preventable weaknesses in Matter's concept of time synchronization and usage or any great risks.

5.4 DISCOVERY IS INHERENTLY INTERESTING, PRACTICAL INVESTIGATIONS RECOMMENDED

We found some potential issues of the device discovery process that requires practical assessment. As pointed out by the Matter specification, the announcement of the VID and PID poses a privacy risk as it allows an eavesdropper to base assumptions on the interiors of a building [30, Chapter 5.2.2.1.3]. Giving away information about the device types used in a facility can provide information about the room layout and number of inhabitants and can be used for espionage. Further-

more, the advertised information can be harmful or embarrassing, for example if the advertisement reveals make and model of a sex toy, which might be displayed by a commissioner during commissionable node discovery [30, Chapter 5.2.2.2.4]. DNS messages optionally contain potentially user-controllable input in form of the 32 bytes TXT record for the Distinguished Name (DN) or device name, probably also referred to by friendly name in the specification [8]. Commissioning applications might display the DN, and therefore we consider it as one of the potential access points for attacks. There is an avoidable privacy risk when manufacturers decide to advertise the field per default. Aside from debugging purposes and the ability to build closed environments with filtered discovery view, we do not see any use of the VID and PID during discovery and argue, that the risks outweigh the benefits. We therefore suggest to require privacy-per-default on any discovery information, such that the user is required to enable advertisement of DN, VID and PID.

The choice of interoperability is passed to manufacturers by allowing commissioning apps to apply arbitrary filters of advertisements, that can be used to create semi-closed environments. Since the vendor ID can be spoofed during device discovery, the commissioner would be required to repeat the check during device attestation verification, because the onboarding payload is not guaranteed to include the vendor ID if a manual pairing code is used [30, Chapter 5.1.1.2]. There is a risk that manufacturers exploit the filter function to cause faulty behavior with devices of other manufacturers, to harm their reputation, which can be done so subtly that it is hard to notice the effect during short-term usage (certification), but will influence user's choice during long-term usage. There is a risk that manufacturers collude against other manufacturers to drive them out of the market. It is questionable whether hostname randomization is suitable to prevent tracking and linkage, because devices are required to advertise their discriminator [30, Chapter 5.4.2.4], which is a fixed value, and is inherently suitable to *discriminate* the device [30, Chapter 4.3.1.3].

It is impossible to distinguish genuine from malicious responses during commissionable node discovery, which the specification claims to be detected when application-layer security fails, and therefore, the only remaining threat would be a DoS attack [30, Chapter 4.3.1.14]. We disagree with this in so far, that application layer security is not guaranteed, especially when considering concerns on passcode secrecy in Section 4.2.4. Malicious replies can be part of an attack chain that breaks application-layer security, and for commissioner discovery, malicious replies can result in a device pairing with a malicious commissioner, because the commissioner has no means to verify the commissioner is genuine. The specification suggests to remedy the DoS attack by physically removing the device [30, Chapter 4.3.1.14]. This can be impossible for the genuine user if the attacker places a malicious device such that the user is not able to trivially locate and access the device, for example when it is installed in the subceiling of a semi-public toilet.

Edge routers such as Thread Border Routers have generally more power over the network than regular nodes because they can drop, modify, and inject new messages. Modification can be performed on plaintext for session establishments and discovery messages, and on ciphertext for secured messages. A malicious edge router has the capability to perform a DoS on any communication between two devices of different communication technologies, because it can drop any message. Furthermore, it is in a significantly easier position for MitM attacks against such two devices, because it is neither required to respond faster than the legitimate node, nor is it required to jam any

8 The specification was written by many authors, so that not all wordings were consistent throughout the document, therefore we sometimes had to guess that two distinct terms might refer to the same thing.

messages. The attacker can abuse the functionality used to enumerate diagnostic services used during operational discovery and commissionable node discovery for reconnaissance purposes. This poses a privacy threat if information such as PID, VID and DN are included, as argued above. The specification claims, that the period of time is of importance, when information is broadcasted [30, Chapter 5.2.2.3.4]. We disagree with this, because the time period is not of concern, the only concern is whether a malicious actor was present at the time of a single broadcast. Due to the security-wise claimed independence of underlying communication channels, it has to be assumed that what has been broadcasted once is potentially known to the adversary. Furthermore, the configurability of commissionable node discovery timeouts by the user is underspecified in so far as that is does not provide requirements on configurable time spans and accessibility of the configuration. There is a risk that many manufacturers will fail to allow easy privacy-preserving configuration due to modern trends of data mining. It yet needs to be assessed whether the on-network attacker can abuse the UDC 'doorbell function' for a DoS or energy depletion attack by consuming a commissioner's resources or depleting its energy.

Summarized, we did not find any direct vulnerabilities during discovery, but discussed in how far the information advertised during device discovery can impact the user's privacy, that devices cannot distinguish genuine from malicious replies, that there is a potentially user-controlled field that attackers might use as access point, that edge routers are in a significantly easier position for MitM attacks, and that discovery poses a risk of DNS-based DoS or energy depletion attacks.

5.5 COMMISSIONING – ON ROGUE COMMISSIONERS AND EVIL TWINS

Matter uses a sophisticated commissioning system, as described in Section 4.2.7. Nevertheless, we identified various weaknesses – most of which in combination with assumptions on UI design and user interaction – which we expect adversaries to exploit in mostly targeted attacks, as described in the following subsections.

5.5.1 THE ROGUE COMMISSIONER ATTACK

When commissionees use commissioner discovery to initiate the commissioning process, the user interface (if present) may display the returned information such as the VID, PID, friendly name, discriminator and extended data[30, Chapter 4.3.3]. When a user selects an administrator to whom they want to commission a device to, they are presented information obtained by DNS advertisements. As previously argued, devices are not able to distinguish between genuine and malicious DNS advertisements. Because of this, attacks similar to the evil twin attack against Wi-Fi [74] are possible in the Dolev-Yao model, which we refer to as the *rogue commissioner attack* from now on. In this attack, an attacker-controlled device is configured such that it advertises the same information as the genuine commissioner it wants to impersonate, to trick the user into initiating commissioning with it. For the attack to work, the rogue commissioner has to reuse the same passcode as the impersonated commissioner, but it can use a distinct set of network credentials, because they are configured during the PASE and require no user interaction. This is possible when none of the

values a user is presented are secured and because there is no mechanism during the PASE that allows the commissionee to verify the commissioner. Many attacks in which the attacker becomes a malicious commissioner, including the rogue commissioner attack, are preventable by mutual authentication during the PASE [9]. The attack is even more devastating when the attacker is able to suppress advertisement by the genuine commissioner (e.g., by jamming, or when a malicious edge router drops the response), because a duplicate entry is avoided and the attack remains unnoticed. Rogue commissioner attacks are not easy to mitigate, because in this case, the wish for security and privacy contradict each other: From a security point of view, the attack can be mitigated by requiring to display a unique identifier of the device, which from a privacy perspective is not desirable, because such practice makes it easier to track devices. We suggest that manufacturers could pass the decision on to users, by allowing them to configure how devices are advertised [10]. All in all, we expect to see rogue commissioner attacks in the future.

5.5.2 DEVICE ATTESTATION VERIFICATION IS OPTIONAL, WHY NOT MAKE IT AN OPT-OUT?

Implementations may choose to ignore DA verification results. Therefore, and because of the missing mutual authentication during the PASE, we derive that the claim, that only Matter-certified devices can communicate with each other in a Matter network (see Section 3.6), is wrong. The claim were true if successful device attestation was mandatory, and as long as no attacks against the DA verification were found [11]. Examples of reasons for failure are: the device type is not certified, the commissionee cannot prove possession of the DAC private key, or problems with the certificate chain. Because of this, the user has no guarantee whether commissioned devices conform to the Matter specification, but will most likely expect them to, because device certification is one of the prominently advertised security features. Allowing non-certified devices into the Matter network is a beneficial feature for developers that wish to integrate their own, uncertified devices into a fabric, and allows Free and Open-Source Software (FOSS) communities into Matter without requiring them to undergo certification. Therefore we want to stress out, that it is reasonable to make DA verification an *opt-out*. By this, Matter can both keep to its claim that in the default setting, only certified devices enter a fabric, while advanced users are able to integrate uncertified applications.

5.5.3 THE EVIL TWIN ATTACK

When the commissioner triggers the commissionee to connect to the operational network, the attacker can trick the commissionee to join an evil twin network [74]. We did not find any mechanisms that allow Matter nodes to detect or prevent the evil twin attack against Wi-Fi [74], where

9 Note though, that *mandatory* mutual authentication will almost certainly reduce contributions by the FOSS community.

10 Security- and privacy-wise good UIs should ensure to sufficiently inform their users about the implications of their decisions, and therefore the Matter code repository could add standard hints which developers can integrate into their implementations.

11 We found no attacks against the the DA verification

the evil twin is a rogue Access Point (AP) that appears to the user as an identical copy of the genuine Wi-Fi AP, tricking the user into connecting to the malicious network. The attack is not limited to commissioning, but might occur during operational use whenever a loss of connection occurred.

The commissioner may delegate the following steps to one of the administrators configured in the device's ACL. It is underspecified, how the administrator is selected, which allows further attack surface if an attacker manages to get in a multi-admin position. The administrator discovers the commissionee and verifies that it can successfully establish a CASE session with the commissionee. At this point, the failure is noticed and the commissionee will resume commissioning starting from the ACL configuration or device discovery, depending on whether the devices support concurrent network connections, and the commissioner aborts after 20 failed attempts [30, Chapter 5.5.1]. We explain in Section 5.7 how the attacker can trick both nodes into connecting to the evil twin network using Matter security features. Therefore, the evil twin attack on Wi-Fi can be used to DoS commissioning without requiring to DoS the operational or commissioning network.

5.5.4 SPOOFING OF THE PAIRING HINT VALUE

There is a risk that the attacker is able to trick a user into opening a malicious website with a commissioner or commissionee (which most likely is a device such as a smartphone or television), even if the attacker is not able to influence the rest of the commissioning to the genuine device. This is because the `CommissioningCustomFlowUrl` is determined using the PID and VID [30, Chapter 5.1.1.3] advertised by the commissionee over the unsecured device discovery (see Section 4.2.6). The attacker might be able to spoof the value such that the advertisement contains a different PID or VID. This becomes problematic when there is at least one abandoned or compromised `CommissioningCustomFlowUrl` included in the DCL, and when the commissioning application does not check whether the DCL entry was revoked. The commissioner is not able to confirm that PID and VID are correct unless the information is explicitly provided by the user, or if the application itself contains other means of restricting valid identifiers. Apart from the security risk for the user when opening the malicious link, the vulnerability may be used to launch a DDoS attack against a manufacturer. However, the risk of a DDoS attack is rather low because the attacker would be required to have compromised a lot of operational networks and trigger commissioning on a lot of device pairs at the same time. Matter can mitigate this risk by adding mandatory information revocation checks to *all* DCL read queries to protect against calls of abandoned URIs [12] and add a quick-response mechanism to the DCL such that when a genuine manufacturer's URI target is compromised, the manufacturer or CSA can immediately revoke the entry and add a fresh entry with another or no URL. This way, the time frame in which an attacker can launch such an attack is significantly reduced. Otherwise, the security of the custom commissioning flow is reduced to the security of the weakest manufacturer, which is a security anti-pattern Matter should avoid.

12 When a manufacturer ceases to exist, the URL may be bought by a malicious party, who uses the website to deliver malicious content. The attack might even go unnoticed if the attacker duplicates the genuine manufacturer's website, but additionally serves malware.

5.5.5 ADDITIONAL THOUGHTS ON COMMISSIONING

During commissioning, commissioners can scan the networks visible to the commissionee [30, Chapter 5.5]. There might be edge-cases, where this behavior is not desirable, e.g., when the commissionee has configured private networks, such as a company Virtual Private Network (VPN) that is required to remain secret. Such a use case might occur when the Matter application was executed on a smartphone or laptop. This is a rather exotic privacy issue, but might be important for some individuals to know, such as agents or military personnel. The reversal used for the fail-safe is not described in the specification, and it is interesting to explore for specific implementations, whether the procedure is secure against side effects such that an attacker is not able to produce a configuration c_2 by fail-safe, that is different from the initial configuration c_1.

5.6 MATTER'S USE OF HTTPS-BASED UPLINKS

One of the reserved URL parameters is reserved to transmit operational passcodes. There is a risk, that a device accidentally submits a static passcode, which must then be considered known against the most powerful malicious actors, as described below, as well as the receiver of the request, which usually is the manufacturer.

Manufacturer-controlled URIs are distributed over the DCL. While the CSA can control, what links are called, the CSA cannot control anything that happens after the call. It can be argued that the CSA should make minimal requirements on the security of the connection, such as use of TLS above a specific version, but not SSL, or HTTPS downgrade prevention. Attacks on transport security pose security and privacy risks, such as MitM attacks and eavesdropping. They could even go as far as integrating any outside calls into the certification model, by registering a dedicated web CA for these purposes, and issuing certificates to ensure, that Matter manufactures conform to security best practices. The benefit would be, that security-aware users can report manufacturers, when they misbehave, and their certificate can be revoked. However, such a procedure can also downgrade security because it has potential to slow down innovation at the manufacturer's side if they have to undergo certification each time before applying an update to the website or resource, and it might cause too much workload for the CSA. A more feasible solution might be to implement nodes such that they operate in an *HTTPS only* mode.

Daily checks for updates benefit security, but contradict privacy, because the destination host of any URLs called by a device are monitored by the Internet Service Provider (ISP), which can link a user and device. Sometimes, they even perform MitM attacks on HTTPS, see Kazakhstan MitM attack as investigated by Raman et al [69]. We must therefore assume, that any HTTPS-only encrypted Matter contents are not secure against nation-state and other remarkably powerful actors. We advise that Matter takes precautions to protect against the uprising trends of mass surveillance and nation-state malware by making it mandatory to use application-layer security with proven cryptographic algorithms.

Our key message is, that although *some* devices might function without the need to establish a potentially unsecured connection to the internet, that the user will only have as much choice as there are companies that meet the user's needs. Any devices that initially or frequently connect to

outside of the local network can pose a security or privacy threat for some individuals, and it would be favorable if users were given choice which risk they are willing or able to take, and which not, including whether the mode of operation requires outside calls, or can be used purely within a local network.

5.7 RECOMMENDATION OF MINIMAL REQUIREMENTS FOR THE UNDERLYING COMMUNICATION CHANNEL

In Section 4.2.9 we briefly discussed that Matter explicitly allows the use of unsecured channels for a limited set of purposes. We argue that this is acceptable because of Matter's intention to be independent of the security of underlying channels. We noticed that manufacturers might choose to use the device passcode to secure the underlying channel, which would introduce significant attack surface and might enable an attacker to obtain a device passcode by this. We recommend that Matter adds recommendations on choosing secure BLE and Wi-Fi protocols and configurations to help non-security aware manufacturers with reducing attack surface, based on the security considerations in Appendix C. Regarding BLE, no version below 5.2 *should* be used and Secure Connection Only Mode (Secure Mode 1 with Security Level 4, FIPS mode) should be enabled during device commissioning. For Wi-Fi, WPA2 or newer should be used. Supporting a wide range of Wi-Fi security protocols poses the risk of downgrade attacks, but since Matter promises independent security, any weaknesses in the underlying communication channels are an acceptable risk unless they are not preventable on application layer. We did not find any mechanisms that allow Matter nodes to detect or prevent the evil twin attack against Wi-Fi (see Section 5.5.3).

For operational networks over Wi-Fi, the most secure type available should be used, which is a double-edged sword: One the one hand, it allows nodes to automatically connect with the highest security protocol available for the network. On the other hand, the attacker can *upgrade* network security to trick a node into connecting to it, as long as the size of the network key is legal for both security protocols. Furthermore, the attacker can target messages of one specific security type to decrease their link quality such that the node is forced into downgrading to a less secure Wi-Fi protocol. The order of security is only implicitly defined by order of listing in the specification, an ambiguity that can be easily resolved.

We advise to continue allowing the use of unsecured channels, but add recommendations of channels to the specification that are widely considered secure. This is because there are potential risks of replay attacks, malicious DNS responses and privacy breaches via advertisements, mostly depending on device-specific implementation details, as discussed in the surrounding paragraphs.

5.8 MULTI-ADMIN

Multi-admin and multi-fabrics allows users to add nodes to multiple fabrics, and assign multiple administrators to a fabric. As briefly discussed at the beginning of Section 4.2.10, the feature is not well-defined such that we base the security analysis on our understanding of the functionality, where we assume that each node can have only one RCA, and therefore, an attacker is not able to trivially commission nodes to fabrics that root in another RCA. Under this assumption, an attacker

that wants to add a node to a fabric must have compromised either of the RCA or ICAs of the user's genuine fabric [13]. We strongly advise that an unambiguous description of the multi-fabrics and multi-admin feature is added to the specification, to avoid manufacturer-specific behavior. We believe there is a considerable risk that an attacker with one-time access to an administrator device or means of target surveillance adds a malicious administrator, and adds all nodes into a fabric under the attackers control, as explained below. The attack scenario mostly applies to targeted attacks, were the attacker interacts with the user in the physical world, and is therefore limited in scalability.

In the following, we analyze the security of the Enhanced Commissioning Method (ECM) and the Basic Commissioning Method (BCM), both of which are available to the Dolev-Yao attacker. During ECM, multiple nodes can be commissioned to a new fabric and administrator at once, using the same dynamic passcode. This feature makes ECM very interesting for an attacker, because it does not require knowledge of any of the devices' passcodes. We distinguish three attack scenarios: (A) ECM is triggered by the genuine user, (B) ECM is triggered on a genuine administrator to which the attacker gained one-time access and (C) ECM is triggered by an exploitation of the administrator application.

Scenario A and C risk, that the attacker is able to obtain the passcode the moment the administrator outputs it, for example by using a long-range camera or directional microphone, and automate input such that the attacker commissions faster with the device(s) than the genuine user. In this case, the attacker would succeed to commission the node to an attacker-controlled administrator. It is undefined how the user is notified in this case, and there is a risk that the user mistakes the attack for a software bug, which might allow the attacker long-term access to the devices. There is also a chance that the user does not notice the attack, if the attacker puts the devices back into ECM mode right after.

In scenario B, the attacker commissions the devices to a new administrator by having one-time access to the current administrator. There are three situations in which this can occur: (1) the attacker borrowed the smartphone in unlocked state and (2) the attacker steals the phone for a limited amount of time and returns it after the attack. In situation (1), social engineering is part of the attack chain. The attacker could enter the victim's house as a friend of a friend at a party, then borrows the smartphone on which the commissioning application is used [14]. If the application is not protected by a password and if the devices are in reach, for example because the attack is launched by a party guest at the victim's house, the attack will potentially go unnoticed unless the user has means to and actively checks the list of fabrics their devices are commissioned to. A mitigation could be to notify the user via a second factor, but if the second factor is an e-mail, there is a chance that access to the second factor is equivalent to access to the unlocked phone, which is quite usual for currently used second factors. In situation (2), the attacker is required to unlock the phone in order to be able to trigger ECM. Multiple techniques have been described for this, such as shoulder-surfing, guessing (e.g., the unlock passcode is the birth date, which is potentially publicly known), via infrared [95], grease patterns, fingerprint faking [55] or bypass face unlock with a photo [90]. From our experience, a committed attacker is able to unlock a foreign phone if at least one encounter with the target and phone in hand were made. We argue, that Matter's security should not depend on the security

13 Though there is a risk that a compromise of the commissioning application could yield comparable results.

14 Possibly by telling the victim they have forgotten their phone and need to look something up on the internet or a map.

of a device lock. Example of an attack scenario: *Buck, the agent, wants to bug Alice's home, but cannot get past the front door and therefore temporarily steals Alice's phone, which they have seen Alice using to open the door. They can unlock the device, because they had shoulder-surfed Alice once while unlocking the phone. Buck wants to access the house multiple times, and therefore commissions Alice's door lock to a device in their control, using ECM. Afterwards, Buck stealthily returns the phone. As an outcome, both Alice and Buck can open and close the front door.*)

Therefore, we strongly advise to require all commissioning applications to support setting a password, PIN or biometric, possibly as an opt-out [15], because it sufficiently mitigates attack scenario B.

In scenario C, the attacker is able to exploit a vulnerability in the administrator application, triggering the current administrator to output the temporary passcode and instruct one or more nodes to enter commissioning mode. The scenario is similar to scenario A, but the ability to perceive the temporary passcode might be limited depending on how the administrator device behaves when it is locked or the screen is turned off. We therefore recommend to prohibit the display of passcodes via notifications on screen lock and deny administrative actions if the screen lock is turned on. The specification recommends to allow commissioning only with some form of physical proximity user interaction to protect against long range camera passcode retrieval and commissioning without device access, however, devices that are inherently subject to physical tampering should not use physical interaction with the device itself to initiate commissioning, but instead use means such that both passcode and the unlock button can be stored safely [30, Chapter 13.6.2]. We argue that any device can be subject to physical tampering, because humans are social beings, and strong and robust security models should not involve blind trust in a subgroup of people, applications, manufacturers or providers. We believe that more research has to be put into designing a secure and easy-to-use multi-admin and multi-fabric commissioning.

BCM risks malicious commissioning by a potentially remote attacker in the Dolev-Yao model the moment the commissioning window is opened if the attacker is in possession of the device passcode. Such a scenario is described in T173. Therefore, the specification recommends to implement BCM only on nodes that can adequately secure the passcode (CM154) [30, Chapter 13.7]. However, we argue that an attacker, who is able to trigger BCM, is also able to trigger ECM, because both require interaction with the current administrator, but BCM additionally requires a passcode as second factor, while for ECM, the only factor is access to the administrator device. Therefore, we suggest that BCM is *more secure*, especially in a social engineering-based attack scenario, unless offline precomputation attacks are involved, where ECM is stronger due to the random password generation.

It is unclear whether an application would be allowed to offer the user choice between ECM and BCM, and whether the attacker would consequently inherit the choice between access by knowledge of the default passcode (BCM) and access by obtaining a new random passcode (ECM). Even worse, an attacker might be able to turn the devices back into commissioning mode to avoid detection. Because the attack does not impact the user's ability to control and use the devices, the attack may go unnoticed and persist for a long time. Besides others, the specification recommends

15 However, as already discussed, commissioners are not required to be Matter-certified, so that the risk is currently not fully preventable for certified commissionable nodes.

to allow the the administrator to view all fabrics for which a device is commissioned [30, Chapter 13.7, CM41]. By this, the user is able to perceive the attack, but active investigation is required.

To sum up, multi-admin provides a convenient means to commission multiple devices, but we see a security risk especially in ECM, in which neither physical interaction with the commissioned devices nor knowledge of their passcodes is required, for attack scenarios where physical proximity to the genuine user is implied.

5.9 IDENTIFIERS

Communication in Matter fabrics requires nodes to use various unique identifiers. We noticed that in some situations, compressed fabric identifiers are used, which risk collision. It has yet to be assessed what the odds and implications of such a collision are.

Administrative Domain Manager (ADM) is used to ensure uniqueness of fabric and node IDs, but because ADM is out-of-scope of the specification, it cannot be assumed that interoperability and uniqueness are guaranteed. And because ADM is undefined behavior per specification, it risks vulnerabilities based on confusion of two identifiers.

Matter defines various reserved IDs. There is a risk that implementations contain flaws in the execution flow, e.g., there is a risk that logic checks are passed in an undefined order, leading to undefined behavior, which could bypass security mechanisms. These flaws would be implementation-specific, however, and do not impact the security of the Matter specification. There is a risk that developers forget to disable test vendor IDs, which would allow an attacker to use these IDs to act as a Matter-certified device, allowing the attacker to pass DA verification. Once a device is commissioned it is equally privileged in the network, regardless of its certification status. If the certification procedure failed to check that test vendor IDs have been disabled, Matter risks that manufacturers start building devices that can interact with Matter devices, but are not certified. Users might not understand the subtle difference between a 'Matter-compatible' (non-certified) and a Matter device (certified). Under the assumption that one of the reasons for certification is quality and security assurance, it is to be expected that cheap eastern counterfeits could harm Matter's network security and privacy, and therefore, we advise that testing that test vendor IDs have been disabled should be a mandatory part of certification procedure.

The range of VIDs and PIDs [30, Chapters 2.5.2, 2.5.3] limits the number of manufacturers and products that can participate in Matter [16]. We therefore assume that Matter is not meant for eternity, because extending the range later on will probably be impossible without deprecating interoperability with older devices. Another strategy is to trim the DCL such that VIDs of companies that ceased to exist can be reused. In 2021, there were about 333.34 million companies operating worldwide according to Clark [20], and 613 IoT platform companies according to Wegner [89], with an 2.4 times increase from 2015 to 2019, but a slight decrease from 2019 to 2021. Wegner believes, that the market has begun to consolidate. If this were the case, we assume that there are sufficient VIDs and PIDs for at least a century, and it probably is fair to agree that even a worst-case scenario would allow longer operation than current IoT platforms.

16 VID and PID are 16-bit numbers so that at most 65,536 vendors with at most 65,536 products each can participate.

Summarized, we are currently not aware of any direct vulnerabilities of identifiers, and therefore focused on suggesting properties of identifiers and their assignment that can potentially cause attack surface. We recommend, that the CSA adds a specification for the ADM mechanism instead of shifting the responsibility of ensuring uniqueness to manufacturers.

5.10 SLEEPY END DEVICES AND UNSECURED PARAMETER EXCHANGES

Because SEDs are equally secured as regular nodes, our focus lies on considering energy depletion and DoS attacks, because SED behavior in general can introduce further attack surface for these two attack types. The specification is unclear about whether SED parameter configuration is possible for the manufacturer or the device administrator. It is therefore unclear whether a malicious second administrator can modify the values. In general, the attacker can deplete the node's energy by increasing the active interval and decreasing the idle interval, and they can slow down polling by increasing the idle interval. Slowing down polling is specifically interesting if the device retrieves sensor data and is configured to issue warnings on specific values or changes, while energy depletion can cause premature end-of-life if the battery is not rechargeable or exchangeable, or temporary DoS. A Non-Matter-specific depletion attack can be performed by continuously messaging the SED to prevent it from switching to Idle mode [30, Chapter 2.9]. During the PASE, the SED parameters are part of the context, preventing the attacker from modifying the values. However, during the CASE and discovery, the SED parameters are transmitted such that any modification of an attacker (e.g., by a malicious edge router) goes unnoticed, because they are sent in unsecured plaintext instead of being signed, hashed or encrypted, allowing the attacker to influence polling and transmission behavior. Furthermore, it is unclear whether a device can switch from being a SED to being a regular node and vice versa, which would lead to undefined behavior for group membership in the non-sleepy node group [30, Chapter 2.5.4.1]. Undefined behavior can cause security problems, but we found it best to examine SED behavior on real devices, which is left for future research. Future revisions of the specification should clarify whether the Matter administrator should clarify whether SED parameters can be configured by the device administrator and how the parameters can be modified. Apart from the concerns about energy depletion and DoS, SEDs are securely defined in Matter such that we do not expect any additional attack surface on them.

To sum up, we noticed that the configurability of SEDs is underspecified, and that the attacker is able to alter SED parameters which are transmitted during discovery or CASE, but is not able to alter SED parameters sent during the PASE. We therefore expect that an attacker is able to influence transmission rates, which could cause a DoS or deplete energy. Therefore we advise that future revisions of Matter ensure that SED parameters are moved to the message part *to-be-signed*.

5.11 OPERATIONAL SESSION ESTABLISHMENT, REPLAY AND RESUMPTION

Operational communication is secured over CASE sessions, as briefly described in Section 4.2.13. An attacker might abuse the asymmetry of the exchange for a DoS: The `Sigma2` message with which the responder replies are computationally more costly than the initial `Sigma1` message due to the cryptographic operations involved. It is possible to replay `Sigma1` messages, and therefore, the

responder might store previous destination IDs to reject replayed messages [30, Chapter 4.13.2.4]. However, this is no official requirement, and therefore, the responsibility is shifted towards the manufacturer. Such a replay protection risks that an attacker DoS attacks a genuine session establishment, when replaying a genuine `Sigma1` message for a device, that later on and within the same epoch wants to establish a new session. On this occasion, session establishment could be denied to the genuine device, because the message is rejected as a false positive. It is left for future research to explore how the SIGMA exchange can be further protected against replay attacks. The Dolev-Yao attacker can therefore deplete computational resources of commissioned nodes and by this attack the availability of a node, if they were able to capture a valid `Sigma1` message with the target node as destination. This impacts the CSA's claim that no further security measures are required to secure communication over Matter (see Section 3.6). Epochs have the benefit of not requiring to distribute revocation lists or check whether the certificate of a communication participant was revoked, but instead shifts the effort to a (semi-)centralized party. As argued in Section 4.2.2, insufficient RNG implementations risk that common secrets such as the `SharedSecret` and `ResumptionID` are repeatedly used, which could result in an attacker being able to resume the previous session of a genuine device. However, the `SharedSecret` is an ECDH-generated secret which can be assumed secure under the assumption that the public key cryptography of a specific implementation is secure, and the `ResumptionID` is protected by encryption with the corresponding `SharedSecret` and can therefore be considered known if the `SharedSecret` is known. For the `destinationId`, the attacker must be in possession of the node's full identity and the current valid IPK of the responder. The attacker must be either in the fabric or have compromised an in-fabric node to obtain the IPK from which they can derive the `destinationId`. Session resumption therefore has the weakest security assumption against an inside attacker, where the entire security relies on the secrecy of the `SharedSecret` if the attacker was able to capture the previous CASE they wish to resume.

In summary, replays of `Sigma1` are currently possible. They cannot be used to hijack sessions or for spoofing, but can be used to to deplete computational resources or DoS genuine communication, depending on whether replay protection is implemented. The security of resumed sessions relies on the secrecy of the `SharedSecret`. We advise manufacturers to ensure, that their devices have good randomness, else they risk that an attacker is able to resume a previous session.

5.12 MESSAGE COUNTERS AND REPLAY ATTACKS

As described in Section 4.2.14, message counters are used as nonces for encryption. It is of great importance that no two distinct messages M, N use the same message counter c with the same encryption key, else it might be possible to (partially) decrypt the message (see Section 2.3.1). A node that does not properly validate that a message was within the expected window of message counters is vulnerable against replay attacks, because it fails to discard messages with invalid message counters. Initializing the message counter in a range from 1 to 2^{28} is privacy-wise a good property and still provides an entropy of almost 2^{32} message counters, before rekeying is required. Because the first authenticated message counter is trusted when a trust-first policy is used, we derive that Matter nodes are vulnerable against replay of control messages after reboot, the impact of which has yet to be examined. The attacker cannot establish sessions by this, because these involve a random from the genuine party, on which subsequent messages have to be adapted based on knowledge of

a secret unknown to the attacker. It might be possible, however, to replay operational messages, by which the attacker might control a device. Implementations that do not implement cache-and-sync might be vulnerable after reboot against replay of any messages. Message counter synchronization can be triggered by entirely filling the group peer state table, on which a node starts deleting the oldest entries. Furthermore, the table is not required to be persisted, and therefore, a reboot can trigger message counter synchronization [30, Chapter 4.16.2].

In conclusion, we recommend that future research focuses on the feasibility and impact of replay attacks, that are based on message counter re-synchronization, due to the potentially high impact, because the attacker might be able to invoke commands.

5.13 STATUS REPORT MESSAGE INJECTION, BYE-BYE SESSION ESTABLISHMENTS

The four unsecured status report messages of the secure channel protocol can be injected into the network by a Dolev-Yao attacker. The status report messages `NO_SHARED_TRUST-ROOTS`, `BUSY` or `INVALID_PARAMETER` can be used to interrupt session establishments between two genuine devices, where `INVALID_PARAMETER` is potentially expected to be received from either initiator and responder in response to either of the `Sigma1`, `Sigma2` or `Sigma3` messages, `NO_SHARED_TRUST-ROOTS` is potentially expected from the responder in answer to a `Sigma1` message, and `BUSY` can be expected from the responder in response to a `Sigma1` or `PBKDFParamRequest` message [30, Chapters 4.13.2.3, 5.5]. These messages allow the Dolev-Yao attacker to abort any CASE or PASE and by this harm availability. Upon receipt of a malicious status report message during a session establishment of two genuine nodes, the receiving party will abort the session establishment, while the other party will try to continue the session establishment, but will eventually fail either due to timeout, or because the party that received the malicious status report message responds with another status report message indicating an error [30, Appendix D.3.1]. There are currently no mitigation strategies in place that can protect the availability of the PASE and CASE against a Dolev-Yao attacker. There is a risk that a node prematurely receives a malicious `SESSION_ESTABLISHMENT_SUCCESS` during a session establishment, which might trick a node into aborting the PASE or CASE with undefined behavior if the implementation fails to properly model the process flow. An attacker can abuse the mechanism of `BUSY` (see Section 4.2.15) to influence the random by discarding unfavorable randoms. This slightly decreases the difficulty of a precomputation attack, and increases the feasibility, respectively, because the attacker may prepare for a smaller subset of randoms. However, as argued in Section 4.2.1, the entropy of one genuine party is sufficient on its own during the PASE. During the CASE, the random is used as the salt during the derivation of the destination identifier and for resumption keys, the former of which is still protected by the IPK and the latter of which are protected by the shared secret. The risk of actual exploitation is therefore very low. `BUSY` can be used to examine the randomness of a commissioner, especially when implementations do not limit the number of retries before session establishment is aborted.

In summary, we have shown how a Dolev-Yao attacker can seriously impact availability of commissioning and operational communication by abusing status report messages to DoS session establishments. Such attacks might be mitigated by temporarily enforcing signing status report messages if a party received a genuine response shortly after it aborted session establishment after receiving a corresponding status report, or if subsequent status report messages of arbitrary type caused ses-

sion establishment interruption, without a follow-up genuine response. Unless the attacker is able to forge the signature, the two genuine parties can continue the session establishment, and ignore injected status reports. However, this method is currently not trivially available, because the corresponding public keys are distributed at later points of the protocol flow, such that adjustments of the PKI and key distribution are required. We suggest that further work is put into developing a mechanism, that is robust against injection of status report messages.

5.14 AMBIGUITIES IN GROUPCAST COMMUNICATION

Groupcast communication in Matter requires unique GIDs. As discussed in Section 5.8 and Section 5.9, it is ambiguous how multi-admin works andthe responsibility of uniqueness of various identifiers is shifted towards the manufacturer. It is unclear, how uniqueness is achieved, especially in scenarios where two administrators concurrently add a new group. There are multiple problems that can occur on when duplicate GIDs are assigned, such as unintended addition of a node into a group, revocation from a group or confusion about permissions defined in a node's ACL. There is a possibility that attackers are able to exploit GID confusion to obtain another group's permissions. Because of this, we recommend that future revisions of Matter include a description how GID uniqueness is guaranteed. It is possible to create logical subgroups, but we found no unambiguous description of them. It is unclear whether subgroups share the message counter domain of the shared security domain or whether they use distinct message counter domains. If the message counter was reused within two distinct subgroups A and B sharing the same operational group key, this causes nonce collision, risking the confidentiality of the payload.

In summary, we found that uniqueness of GIDs is not guaranteed and can cause serious issues with permissions and memberships, and that subgroups are underspecified. We strongly recommend to add precise descriptions on how unique GIDs are guaranteed in a multi-admin setting, where concurrent assignments can be made, and suggest to clarify whether a subgroup shares the message counter window with its parent group to avoid confidentiality breaches caused by misinterpretations of subgroup behavior.

5.15 EPOCHS AND REVOCATION

The epoch keys used to derive operational group keys are updated and revoked by the group key administrator [30, Chapter 4.15.3.2], and if an attacker gained the ability to invoke either of the two commands, the security of the group is compromised. Epoch keys are generally used to ban nodes temporarily or permanently from group communication without requiring other nodes to re-establish a new group session. From a cryptographic perspective, the exclusion is consummated the moment all epoch keys that were known by the excluded node are removed from group state either by the regular rotation or out of sequence replacement of all epoch keys, for example by resetting group security using the `Admin Refresh` command [30, Chapter 4.15.3.5]. Because the latter option can be expected to cause communication errors for genuine nodes due to the difficulty to update all nodes synchronously, it is very likely that less mature implementations will use the former option to exclude nodes from communication, and might instruct the banned node to delete

its group state and keys. A malicious node must be expected to keep the epoch keys even after instruction to remove them, and – from a cryptographic perspective – can continue interaction with the group until the keys were completely rotated through. Because of the contradicting requirements for fast revocation and energy-preserving behavior, it is not trivial to determine a suitable rotation rate. Fast rotation rates with short epochs allow faster exclusion from communication, but are computationally more demanding. Implementations risk privacy breaches if they fail to successfully explain to the user how exclusion from group communication works and that the process might take several weeks, from a cryptographic perspective. Instead, clients might update the UI such that the revoked node is hidden, tricking the user into believing that the group communication is secure against the excluded node, while in reality, the node can continue decrypting and sending messages until the keys have been rotated through. In the following, we describe a scenario in which a user's privacy can be seriously breached:

Assume that Alice, who is a famous actor, throws a party and allows party guest Eve, who is an unknown journalist, to connect one of his devices to one of her Matter groups to stream music, in this example the group used for media streams and output. Furthermore, she has provided him with the Wi-Fi passkey, like she would have with every other guest of the house. Because Alice is security-unaware, she uses no network segregation. Alice removes Eve's device from the group when he is leaving, and Eve's device is not displayed anymore in her controller's view of group members. The following days, Alice is streaming media that can reveal sensitive information about her, such as adult content. She assumes that the information is securely communicated via the group. Eve, however, has stored the epoch keys and operational group keys and has placed a hidden device to monitor network traffic. In this scenario, Eve is able to compromise Alice's privacy and leak very sensitive information to the magazine he is working for. In worst case, he is still able to control her media devices, when group membership is assumed by possession of the operational group key, which can have a serious impact on Alice's psyche. For example, Eve could play horror-themed media to perform a DoS on Alice, the human.

From the author's experience, Alice's behavior and trust is commonly found in people. Alice and Eve could be relatives, friends, room mates, neighbors, coworkers, distant acquaintances or even strangers. The motives can range from personal over business or criminal to political motivations. Eve could be an abusive partner, a co-worker, blackmailer, or have other motives to spy on Alice. We therefore suggest that implementations ensure to unambiguously communicate to their users how group membership revocation is handled, which can only be guaranteed if the behavior is defined in the specification. Since groupcast communication uses bidirectional shared secrets, malicious group members cannot be prevented from reusing nonces to reveal communication to the network in a less obvious fashion (see key reuse attack in Section 2.3.1). In general, a compromised group member compromises the confidentiality of the entire group, and therefore group communication is only as confidential as is the least secure group member secure. Furthermore, a group is only secure as long as the group administrator cannot be impersonated.

To sum up, we found that revocation of group members is insufficiently enforced from a cryptographic perspective, allowing an adversary to remain in the network for a long period time, which can be several weeks depending on how often keys are rotated. We therefore strongly recommend that Matter aids minimum and maximum intervals for key rotation, along with recommendations for several use cases, and ensure that users are sufficiently informed on the exact point of time when a group member is fully removed from a group from a cryptographic perspective.

5.16 REMARKS ON COUNTERFEITING

The Matter specification addresses a fear of counterfeits by various means. Manufacturers should control the number of DACs issued under their VID to protect against counterfeits produced by a worker at the factory or programming location and have unique and authorized credentials [30, Chapter 13.7, T23/CM24]. However, keeping track of the numbers alone does not allow to distinguish counterfeits from genuine devices and does not protect against extracted and cloned DACs and key pairs. Note that the DAC monitoring process itself can be targeted as well and therefore, manufacturers could use digital watermarks to prevent counterfeiting, as described by Abdel-Hamid et al [1]. Another option, however, contradicting the privacy-preserving nature of Matter, is to deploy plausibility checks for DAC validation requests. This could, for example, detect two subsequent validation requests for the same DAC from two different countries, with too few time in between as that the device could have traveled such a long distance, indicating a counterfeit. However, users with VPNs could cause false positives, and the implausibility can therefore be reasonably denied.

In summary, we assume that we will see counterfeits of Matter and solving the issue is hard or even impossible and will require future research to mitigate feasible loopholes.

5.17 ON THE SECURITY OF CERTIFICATES AND KEYS

We noticed that the specification recommends that node operational private keys should not leave the device, but it would not be a violation if a manufacturer chose to store all private keys in a cloud, as long as he was able to argue that the keys were protected from unauthorized access [30, Chapter 6.4.10], which would greatly increase attack surface due to the transmission involved and replicated storage. Furthermore, we noticed that 'authorized access' is ill-defined, because it is ambiguous whether access by the manufacturer, who might argue that the keys are required for maintenance, is allowed. We are currently not aware of any scenario where it makes sense for a device to transmit its private key to another entity, and therefore recommend that the next revision of the specification forbids private key exfiltration in general, with explicit exceptions, if reasonable at all. All nodes should protect the confidentiality of private keys [30, Chapters 13.6.1, 13.7 CM77], but it is not described how to do it. We advise that the certification procedure ensures that private keys cannot be extracted via software or hardware access by simple and known means. NOCs allow to specify the context in which a key is used in the key usage extension, for example, for signatures, key signing or data signing [30, Chapter 6.5.11.2]. The feature mitigates attacks that base upon confusion what step of the procedure is currently performed, similarly to the `FixedInfo` used in Matter's KDFs (see Section 4.2.2).

In summary, for further improvements on key security we suggest that Matter continues to add best practice examples that enable manufacturers to securely store key material, and to explicitly forbid key exfiltration unless for underlying and white-listed reasons for doing so.

5.18 ACCESS CONTROL

Matter's use of ACLs allows users flexible configuration of arbitrary device combinations. In the following, we discuss the implications of Matter's access control.

5.18.1 ADMINISTRATIVE PRIVILEGES AND EVENT MONITORING

Access control ensures that only privileged nodes can access specific assets. Because the administrative access privilege is implicitly granted to the administrative subject during commissioning [30, Chapter 6.6.2], it is of great importance that adversaries are not able to maliciously commission genuine devices, since this would grant them administrative power over them (see Section 4.2.7). Administrative access might also be possible by knowledge of only the `I2RKey` instead of both keys, because this key is used by the commissioner after the PASE and during commissioning to encrypt messages. The attacker does not require knowledge of the `R2IKey` unless they required read-access.

Removing the last subject in an access control entry risks to produce a wildcard, which grants access to any subject. We therefore advise that the certification procedure ensures, possibly by following a routine or fuzz-testing, that no ACL operations result in accidental wildcards, which would break many of the security assumptions in Matter against malicious nodes in the fabric. Furthermore, access control extensions should always be tested for security, because they can potentially lower overall device security.

Matter's event-based logging system allows network administrators to deploy network-based Intrusion Detection Systems (IDS) to identify irregular ACL changes and other events. It has to be stressed out that this development greatly helps improving IoT security from a user perspective, because it allows to build sophisticated monitoring systems around the devices, as further elaborated in Section 5.19.4.

5.18.2 GROUP-WIDE PERMISSIONS

For group-wide permissions, where reachability to all nodes cannot be guaranteed, an attacker might jam the connection between an administrator and target device to prevent the administrator from updating the target's ACL. For malicious edge routers, the attack is quite easy to perform, because they are in an inherent middle-man position that allows them to drop packages. There is a risk that the user is not well-informed upon failures of updating the ACL for single nodes in a group, because it is not well-defined how the administrator is required to notify the user. We were not able to determine by specification review whether long disconnects or commissioner reboots before successful ACL update of all nodes could cause inconsistent ACL entries in group management, and recommend future researchers to test for inconsistencies. Furthermore, malicious administrators, including those who are not the RCA, can revoke privileges of other administrators, including administrative rights [30, Chapter 6.6.4]. This can remain unnoticed until the user tries to perform an action for which the revoked privilege is required. When the user is not able to perform the action, they will possibly factory reset the device to regain administrative privileges.

5.18.3 BRIDGES, CAREFUL UI DESIGN REQUIRED (PART I)

Administrators grant privileges to bridges instead of single bridged devices. This behavior has potential to cause confusion in administrative applications, for example if the view of device settings does not distinguish and properly arrange bridged devices, so that the user is mislead into believing that he granted the privilege to only one device, when instead multiple devices were implicitly granted the privilege. There is a risk that UIs serve outdated privilege views for the other bridged devices, leading the user to false security assumptions. We therefore recommend that future versions of the specification require ACL setting views to always query up-to-date ACL information directly from the administered nodes, and that bridged devices must be visually grouped together such that it is unambiguously clear, that they form a group. Furthermore, the user must be sufficiently informed if the modification of access rights for one device will change the access rights for another device. We recommend that the user per default retrieves a notification each time they configured a bridge or bridged device, but can disable the notification in the settings of the administrative application. Another possibility is to require that bridges ensure that only a specific device is granted specific privileges, but since the communication between a bridge and a bridged device can be entirely unsecured and arbitrarily structured, a bridge cannot give any guarantees that only the specific device is granted a specific privilege.

5.19 DATA AND INTERACTION MODEL

5.19.1 TIMED INTERACTIONS, THE FINAL ENEMY OF REPLAY ATTACKS

Matter uses timed interactions to protect against replay attacks, as described in Section 4.2.19. We noticed that the specification is unclear about whether an arbitrary number of actions is allowed during the interval, and whether multiple intervals of timed interactions may overlap. If this were the case, timed interactions would not fully prevent replay attacks but instead only narrow the time window in which the attack is possible. Consider an attack scenario where a genuine user leaves the house. The attacker would be able to open the door, pass through it and close it right after the genuine user has left the house if commands can be replayed an arbitrary number of times and timed interactions were allowed to overlap. Overlapping of timed interactions is not required for commands that toggle a behavior, such as a light switch.

Because such behavior has potential to allow an attacker to control the outcome of a command [17], we recommend that the specification removes ambiguities of how timed interactions behave, such that a timed interaction allows only invocation of exactly one instance of a command, unless the use case scenario required fine-grained time-sensitive interaction. Note that replay attacks are additionally prevented by other cryptographic means described in Section 4.2.1 and Section 4.2.14, and the attack scenario only applies if the attacker was able to overcome these protective mechanisms.

To wrap up, we consider timed interactions to greatly increase Matter's security, and describe the loopholes left that might be exploited by an attacker.

17 Because users might invoke commands remotely, they might not be able to notice the replay attack.

5.19.2 LISTS AND POTENTIAL MEMORY EXHAUSTION

We reviewed the data types used in Matter and noticed that the data type *list* is not constrained in size with up to 65,535 entries [30, Chapter 7.18.1.9]. An attacker might be able to DoS a device by sending a list that exhausts the target's resources, especially when the list that is sent results in the device trying to create another list from it, that requires significantly more disk space or memory. Manufacturers should ensure, that asymmetric workload is avoided, especially when the data is received from non-administrator nodes.

5.19.3 FACTORY RESETS

The exact set of data to be persisted during factory resets is not described in the specification. We can derive, that some of the cryptographic material such as the DAC or CD must persist, but we advise future revisions of the specification to explicitly list the data that persists across factory resets to remove ambiguities. Furthermore, we recommend that devices must be reset to a configuration that is secure-by-design, even if that means that the device is reset to a different configuration than it had when it left factory, if the initial configuration turned out to be vulnerable. Therefore, security updates should be able to update the configuration which is applied to devices that are factory-reset.

5.19.4 EVENTS AND NETWORK-BASED MONITORING

Events in Matter, which are similar to log entries, have a timestamp, which is either system time or epoch time. It is unclear how an observer distinguishes system time from epoch time, and whether an attacker could try to exploit confusion about the time provided, but since every node uses monotonically increasing event numbers that are preserved across reboots, any confusions arising from this will be implementation-specific. The handling of priority levels for events prevents the removal of critical event by producing a large number of uncritical event records, which is security-wise a beneficial feature, because it makes log rotation to remove traces of an attack harder. However, we noticed that the specification omitted a description of log bootstrapping, which risks that a manufacturer implements a first-come first-serve strategy. For example, if a device could buffer 1,000 entries, and the first 1,000 entries are info entries, the device might fail to buffer critical records. As a result of the underspecification, the improvement might actually backfire and degrade security for some implementations. We therefore suggest that a future revision of the specification includes a description of how many log entries of each priority level a device must be able to minimally cache. Such a requirement may contain both absolute and relative figures. Since Matter runs over IP, network-based security solutions already used to monitor interaction between desktop computers such as an IDS can be used to further improve network security, and industry-proven monitoring systems could subscribe to events, so that companies could afford to build dashboards around their devices. This is a great improvement towards the security of interconnected IoT devices.

In summary, we appreciate the use of events and expect them to help monitoring, but recommend to add descriptions to secure the handling of priority levels such that critical records are not

erroneously suppressed, when manufacturers fail to see that they are implicitly required to reserve buffer for each priority level.

5.20 bridges, careful ui design required (part ii)

A bridge plays a middle-man position such that it translates Matter actions to Non-Matter actions, and vice versa. The bridge has a very powerful role in the network, because once it is commissioned, it can add arbitrary numbers of Non-Matter devices to the network, or remove them, a power which the fabric administrator does not have over bridged devices, unless the bridge itself passed its power to the administrator. A compromised bridge directly risks control over the bridged devices due to its inherent middle-man position. As already described in Section 4.2.18, other devices can define access rules only per-node and not per-endpoint, and therefore, other devices must allow access for either none or all of the bridged devices and the bridge. It is very important that the user is sufficiently informed about the limitation, so that misconceptions of the security concept can be avoided.

There is a significant risk that the attacker targets a weakly secured bridged device and uses it as an entry point to send messages into the fabric, and invoke commands. We advise that a bridge should whitelist all commands a bridged device may invoke, to prevent bridged devices from invoking commands they are not expected to. Furthermore, we suggest that the bridge should emulate the ACL of each bridged device, so that bridged devices do not share one ACL. By this, only the bridge itself instead of all bridged devices would be required to be granted the union of all permissions required by its bridged devices. Since we cannot make any assumptions on the security of bridged devices, it is inherently impossible to prevent one bridged device from spoofing another. In the worst-case scenario, bridged devices offer no security at all and communicate unsecured. But by taking precautions, the effort required from an attacker to perform the attack can be increased. Matter-to-Non-Matter communication risks eavesdropping, and therefore we suggest that administrative applications issue warnings when a Matter device transmits sensitive data to a Non-Matter device via a bridge. We recommend that Matter requires bridges to use the most secure communication channel available for its communication with bridged devices, because otherwise, bridges would be allowed to use an unsecured channel if the bridged device offered both unsecured and secured communication.

In theory, a bridge is able to add Matter devices to the fabric via bridging, which is explicitly forbidden by the specification [30, Chapter 9.12.1], but technically it cannot be prevented. There is a risk that a user mistakes a Matter device for a bridged device, if the UI does not unambiguously distinguish the two types. This is a realistic scenario, because administrators are not required to indicate that a device is bridged [30, Chapter 9.12.12]. On this occasion, the actual security of the connection is significantly lower than the security expected by the user. In this setting, the attacker is required to compromise a bridge or infiltrate one into the fabric. There are three distinct scenarios: (A) the bridged device is connected to the fabric both Matter-native and via bridging, (B) the bridged device is only connected via bridging and (C) bridging is used to spoof a device. Scenario A leads to undefined behavior, such as confusing visualization. For example, the device might be displayed two times, in potentially nondeterministic ordering and indistinguishable appearance. The user might consider this to be a harmless software bug and send commands by arbitrarily choosing

one of the devices at random. Depending on the number of devices in the network, the user might not even notice the duplication. Scenario B can occur if the attacker was able to decommission the devices, for example, by factory-resetting them, or the attacker could add a device that was decommissioned before, in the hope that the user has forgotten this in the meantime, and uses it once he sees it in the administrator application. In scenario C, the attacker exploits the missing indication of bridging to spoof a device using the compromised bridge. In all three scenarios, the attacker might exploit the confusion to trick a user into granting permissions or invoking commands to a device or via a channel they did not intend to.

In conclusion, we appreciate the introduction of bridges, because they allow for full interoperability with any device. We acknowledge, that due to the conflict between interoperability and security, security will almost always be necessarily decreased, which can be considered acceptable, and the choice passed on to users. To improve security of bridging, we suggest to require all administrators to unambiguously mark bridged devices such that they are visually (and for screen readers) distinguishable, and provide information to the user about the security implications of using a bridge. Furthermore, we argue that bridges should be implemented with an equivalent security level as controllers, due to the high potential impact.

5.21 SUBSCRIPTIONS AND PROXIES

Because the specification recommends to commission proxies into all fabrics in a multi-fabric setting [30, Chapter 9.15.8.3], it is thinkable that manufacturers will design administrative features such that proxies are automatically commissioned to all fabrics that root in an RCA. Because of this, proxies might be an interesting target for the adversary. A malicious proxy could exploit the system by subscribing to a node such that a single and genuine non-proxy subscriber gets evicted and the proxy becomes the only subscriber of the source. By this, the attacker gains the ability to drop messages. For other security consequences we can only guess, because subscriptions and proxies are underspecified on a technical level. We assume that the feature is yet in development and will be finalized in later revisions. This conclusion also stems from the fact that Matter seems to allow subscriptions over a PASE [30, Chapter 2.11.2], a process which is entirely underspecified. In general, we can differentiate two cases: (1) the proxy forwards secured messages from the source to subscribers and (2) the proxy extracts the message content received from the source, and encrypts it again using its own cryptographic materials to send it to subscribers. Scenario 1 does not introduce any potential weaknesses apart from the malicious proxy's ability to drop messages. In scenario 2, the proxy is additionally able to alter and inject messages. A malicious proxy in either settings has the capabilities to trigger false alarms, for example, by dropping vital signs, whose omission trigger an alarm and bypass monitoring, e.g., by dropping one or more specific sensor measurements that are below or above the accepted threshold. Scenario 2 is more disastrous in this case, because the attacker is able to inject false values, for example, to cause false alarms by faking hazardous measurements. Trivially deducible, a malicious proxy can fully DoS subscriptions of a source by dropping all messages.

All in all, it is currently not possible to analyze the security of subscriptions and proxies in Matter, especially because it is not specified how subscription messages are secured. Because subscriptions currently allow arbitrary implementations by manufacturers, it must be expected that

some manufacturers will introduce implementation-specific weaknesses. We advise, that Matter finalizes the technical specification of subscription proxies following scenario 1, because it obviously provides the most security, and suggest that future versions of the specification explicitly explain subscriptions and proxies on a technical level, to remove ambiguities.

5.22 THE GENERAL DIAGNOSTICS CLUSTER

The diagnostics cluster offers features that allow enhanced monitoring and forensics of Matter devices using standardized means, in contrast to 'legacy' Non-Matter IoT for which monitoring and forensics was problematic due to the high heterogeneity. This increases the set of options and reduces costs. We do not expect any cluster-specific vulnerabilities.

5.23 SECURITY CONSIDERATIONS OF UPDATES

Checking for updates on a daily basis, as recommended by the specification, is security-wise a good advice, because in practice, daily updates are a common trade-off between the desire to update as frequently and early as possible, and the desire to reduce computational workload. However, we are missing a requirement on over which period manufacturers should offer security updates at minimum, which we advise to add to the specification to avoid premature obsolescence. Apart from Matter-specified updates, Matter allows vendor-specific legacy out-of-band methods that in worst case implement no security at all, which can result in compromised firmware. Legacy update methods are therefore a potential entry point into a fabric. The attacker could try to compromise the runtime service discovery, which is used to discover OTA providers, to control which node takes the role of the OTA provider. When a (native) update is available, the requesting node is provided a Uniform Resource Identifier (URI) that points to the downloadable image. It is unclear whether the use of HTTP without TLS, which is forbidden for Matter-specific behavior [30, Chapter 11.19.6.4], is also forbidden for vendor-specific update methods. We suggest that this ambiguity is clarified in the next specification version, and that the certification procedure includes security assessments of vendor-specific update methods.

The header of the software image file contains general information such as the VID and PID to which the update applies, a payload size, the image digest and an URL for release notes [30, Chapter 11.20.2]. OTA providers may cache updates, and serve them from cache if they have the correct checksum, checksum algorithm and file size [30, Chapter 11.19.3.3]. The software image file is integrity-protected and authenticated. The checksum provides integrity protection in so far as that it protects against accidental changes such as bit rot, but is not suitable to protect against malicious manipulation, because it provides neither authentication nor non-repudiation. For this, a signature is calculated over the software image. It is not explicitly stated in the specification, but manufacturers can significantly reduce computational costs of signature creation and verification by computing the signature over the software image header instead of the entire update. This is possible because the header contains a cryptographically secure checksum (image digest) and file length, which are sufficient to confirm that the update has not been manipulated. We believe this is the intended way, because otherwise, the checksum could have been omitted, since the vendor

signature already confirms the integrity and origin of the update. An attacker might try to install a compromised certificate on a requestor that is used to verify the authenticity of updates, to trick it into installing malicious updates. In-field upgraded devices have the highest risk, because they must receive the certificate over the internet. We recommend that manufacturers install the certificate used to sign software updates, or a certificate used to sign trustworthy certificates, in the factory data and never update them over-the-air. Verification of authenticity and integrity of software images is mandatory for OTA requestors.

Even though there is a risk of future downgrade attacks due to the version negotiation against the Bulk Data Exchange (BDX) protocol, over which non-legacy updates are served, it can be argued that the session is still secured over PASE and CASE, and if either of them turned out to be insecure, Matter would be completely compromised. Therefore, risking a BDX downgrade attack appears to be an acceptable risk.

It is not well-defined how non-compliant [18] or under-test updates should be handled by end devices. Non-compliant updates risk reducing a device's security if their non-compliance is related to a security feature. We therefore recommend that non-compliant updates are held back until their compliance has been confirmed to reduce the risk of accidentally downgrading security, while ensuring that security patches can be applied on the fast-track. Devices can distinguish urgent updates because announcement reasons for updates are provided. This helps devices to decide on the how and when to apply an update.

The roll-out of non-standard software images should be considered cautiously when nation-wide images are served, because some countries have restrictions on how encryption is used [39]. In 2017, Pell and Soghoian [68] described the impact of deliberately employed security downgrades for the purpose of cellphone surveillance on national security and consumer privacy, and argue that such policies are counterproductive towards strengthening security. Of course, non-standard software images can also be used for valid purposes such as split testing, but we argue that the CSA will play an important role in then future in terms of preventing the abuse of non-standard software images for the purpose of surveillance.

5.24 IN-FIELD UPGRADES TO MATTER

In-field upgrades risk exposure of cryptographic material if not securely implemented. It is not specified how the transmission of the device's CD, DAC and PAKE values is done and whether the in-field upgrade process itself is part of the Matter certification. Furthermore, it is unclear whether it may be entirely unsecured. It is there currently impossible to evaluate in how far the security assumptions of an in-field upgraded Matter device differ from a native Matter device. Security differences may arise, for example, from the device's pre-dating history or from how the PAKE values are distributed.

In summary, in-field upgrades are a great means to improve interoperability. However, they also have a great potential to entirely compromise a device if the distribution of cryptographic material is not sufficiently secured. We advise that the CSA defines minimum security requirements for the in-field upgrade to Matter, to ensure that firmware and cryptographic material are securely

18 Note that by non-compliant, we also refer to updates, for which compliance has not yet been confirmed.

transmitted. Insecure in-field upgraded devices pose the risk of becoming the entry point to Matter networks, which otherwise are potentially secure.

5.25 SOME ADDITIONAL THOUGHTS

Based on the above considerations, we conclude that controllers, edge routers and bridges pose the greatest risk, and therefore suggest that this should be reflected by applying higher security and testing requirements on these devices. The CSA might consider offering certifications in multiple levels, so that security-, privacy- and usability-wise, the more sophisticated devices are honored, which could incentivize manufacturers and be used to inform users [19]. Because of Matter's interoperability, all device manufacturers will suffer in reputation if too many manufacturers offer not sufficiently secured devices, because people tend to contextually generalize, especially when dealing with unfamiliar situations, see Dekker et al [32].

Furthermore, we hope to see more restrictions and details on how sensitive information and cryptographic key material especially is securely stored on devices. Based on the above considerations, we derived a test catalog that can be found in Appendix E. The catalog is, of course, by no means complete, can be used to mitigate most of the weaknesses described in this and the previous chapter.

5.26 POSITIVE REMARKS

In order to shift the balance back to making the point that we consider Matter to be a very sophisticated and potentially secure protocol, we decided to dedicate an entire section to point out the positive features we found during our deep dive into Matter's specification, based on personal opinion and experience.

Matter uses strong cryptographic primitives and refers to strong implementations in accordance to national guidelines, such as strong, state-of-the-art cryptographic hashes, MACs, authenticated encryption algorithms, ECC and TRNGs, which will be secure for a very long time, because it has been designed such that only implementation-specific weak RNGs should be able to break it. Secondly, we consider it to be a good idea to focus on designing a home automation standard such that it does not rely on the security of the underlying channel, but instead becomes independent of it, in combination with a focus on interoperability. By out-sourcing the security effort to a centralized entity, a lot of redundant work can be removed, and powers can instead be channeled to improve security for every participant, which we expect will greatly help to make offensive security more scalable. We greatly appreciate the use of public key cryptography, because it offers flexibility in scenarios where arbitrary nodes enter and leave a network. Furthermore, we were glad to see that PAA certificates do not root in a single point of trust, but can instead be revoked individually without affecting certificates of other manufacturers. We appreciate the effort to minimize data and did not find any contradictory information, because the few information left is easily defended

19 Compare QS certifications in agriculture and food supply chain: https://www.q-s.de/en/ [Last accessed: 2022-12-11]

by usability concerns, if they were omitted. We were also happy to see that SEDs follow the same security rules as any other device, making them comparably secure. Another positive aspect to be pointed out is the use of epochs. Even though we had a rather large discussion on them, we believe that they are the right choice of all the state-of-the-art techniques, because they allow flexible revocation. Any of the weaknesses discussed are fixable or at least mitigatable by manufacturers. We greatly appreciate the use of ACLs, because they allow fine-grained and extensible permission handling. Furthermore, we are happy to see that Matter advocates for daily security updates for IoT, has an open-source policy that allows manufacturers to share secure solutions, and openly accessible specification that allows security researchers to review it.

All in all, we conclude that Matter will greatly improve and contribute to the security of IoT devices.

5.27 FUTURE RESEARCH

Naturally, a master thesis cannot fully cover an entire security analysis of a large protocol as Matter. We hereby suggest future research that can be done to analyze the security of Matter, and by this, contribute towards secure IoT networks. Future research can focus on determining potential attacks that result from a weak TRNG-seeded DRBG, as well as conduct studies on devices samples to assess the randomness of Matter devices in practice, which can be regarded as a validation of the CSA's promise to test security and implementation requirements during certification. We encourage future researchers to analyze the use of ECC and public key cryptography in Matter, evaluate its PKI, to validate whether implementations based on RFC 5280 and SEC1 are considered secure, and whether Matter complies with them. Furthermore, Matter needs to be checked against any security considerations mentioned in these guidelines and known weaknesses. Future research could confirm, that using a counter generation for AES-CCM in AES-CTR is secure. Regarding SPAKE2+, we strongly hope that further security analyses will come up in the following years that also elaborate the subtle differences between the original version, the Taubert and Wood version and the Matter implementation. Such a work could include precise calculations on the feasibility of an offline precomputation attack against Matter's PASE, assuming a passcode entropy of roughly 26.575 bits, or a more precise value. Generally, future research could try to determine in what aspects exactly the security of Matter might rely on the underlying channel, including security features such as authenticity and repudiation [20]. Regarding time and certificate validity, future research could try to find weaknesses in Matter's time synchronization, to influence certificate validity, or to perform a DoS attack. In particular, interesting research targets are how the Current Time or Last Known Good UTC Time of a device could be remotely manipulated, including the selection of which is currently in use, and the security impact this could cause. Future research should focus on inventing passcode mechanisms such that passcodes of resold devices are no less secure than the initial passcode, while preserving user privacy and ease of use. Furthermore, it could be assessed whether a trivial passcode such as 00000000 weakens the security of SPAKE2+. Future research can explore energy depletion and DoS attacks against SEDs. From a developer perspective, it would be very helpful to be handed a pragmatic list of do's and don'ts specific for the chosen platform in form of

20 Matter promises independence, but has only ever explicitly mentioned confidentiality and integrity in this context.

a cheatsheet. We therefore suggest that research projects could focus on creating such cheatsheets, as doing the work once for all enterprises has far better scalability than every developer team doing this on their own. And last but not least, we recommend to validate or refute our claims in practice on real devices.

5.28 LIMITATIONS

We did not evaluate the public key cryptography, use of ECC, the DCL and PKI in general, on commissioning over Wi-Fi Soft-AP and firmware security. Due to the limited time frame the verification on an example device was not possible and can be objective in future research.

6 CONCLUSION

The opportunity we have is to build a secure, intelligent platform that solves some of the world's greatest problems at scale. That's what's possible with hundreds of billions of connections and the capabilities that we can deliver together.

— CHUCK ROBBINS

The goal of this work was to answer the question of how secure Matter is by examining the various systematic options currently used by security researchers and ultimately applying a subset of them to Matter. This allowed us to shed light on the status quo of Matter's security and privacy, as well as dangerous deviations that manufacturers could implement, and ambiguities as well as loopholes that manufacturers could find in the specification that could undermine security.

For being able to perform the security analysis, we first had to read through thousands of pages of documentations and specifications (for Matter alone over thousand pages, which equaled a month of work for reading and making annotations), and incorporated our newly gained knowledge into a profound introduction into Matter, which can be read in Chapter 3. For our security analysis, we created a threat model and attack graphs based on the recommendations of multiple authors, performed a threat, vulnerability and risk analysis, and eventually contributed a classical review, which is the major contribution towards answering our research questions. This is probably attributed to the fact that Matter, as a young and recently released IoT standard, has not yet any standard literature, but instead partially contradicting or ambiguous documentation[1], which we did not want to rely on, but instead where required to contribute to literature about Matter, to allow security analysts to take a short-cut and build more systematic models upon our work. Additionally, we extracted the security and privacy claims that were made about Matter by the CSA, for each of which we assessed whether the claim was fulfilled or not. Furthermore, we created working Tamarin theories for Matter's PAKE and SIGMA exchange during the CASE, as well as plain SPAKE2+ that can be used in future research for symbolic verification. Our full methodology is described in Chapter 1, while the threat model, security analysis and a comment on the symbolic verification were included in Section 4.1.

We were able to confirm, that Matter completely fulfills most of the 35 claims we looked at, and all but one claim at least partially (see Appendix A for a summary). Furthermore, we found no direct vulnerabilities, but determined various underspecifications and ambiguities, discussed the circumstances under which we expect Matter's security model to expose weaknesses, provided attack scenarios and determined which parts of the specification risk security-wise significant deviations when being implemented by manufacturers. The discussion on this can be found in Chapter 5. We abstain from repeating or summarizing most of our findings to avoid oversimplification and to paint false pictures, and expect the reader to refer to the corresponding sections, instead. The se-

1 We want to stress out, that none of the documentations we found were bad or wrong, just that they were not suitable for our purposes, and sometimes too old in so far as that terminology pre-release seemed to have changed a couple of times, or is manufacturer-specific.

curity analysis was mostly done prior to release of the first Matter devices. Therefore, our work is based on theoretical assumptions, and we will continue to use our gained knowledge on practical devices in the near future. Most significantly, we discussed why Matter offers a remarkably strong cryptography, and why we do not expect to see any attacks on it soon, while also commenting on the integration of the young and yet-to-be-proven and slightly-modified SPAKE2+ protocol. We found that most of Matter's security depends on passcode secrecy and the implementation of a good RNG, discussed under which circumstances replay attacks are possible, and when not, and what the expected impact will be. We discussed why Matter inherits some of the weaknesses of its underlying channels, and thoroughly examined what a Dolev-Yao attacker is able to do with unsecured messages or the unsecured parts of messages, such as modifying SED parameters and injecting status report messages, that can cause DoS. Furthermore, we showed that Matter inherits Wi-Fi's weakness of evil twin attacks and introduced a very similar attack that we refer to by *rogue commissioner attack*, which can only be mitigated by passcode secrecy. We showed why Matter's multi-admin and multi-fabrics feature introduces additional risks for targeted attacks, especially focusing on why we expect that we will see social engineering-based attack, that will leverage ECM to commission all of the victim's devices to a malicious multi-admin. Furthermore, we had some concerns about attackers being able to spoof discovery messages such that commissionees are pointed to malicious URLs in the DCL, which is possible when an adversary takes control over a manufacturer website or buys an expired website of a vanished company. Privacy-wise, we express concerns that Matter's group communication using epochs is yet underspecified in so far, as that revocation of group members is not yet cleanly done from cryptographic perspective such that the revocation process can take several weeks until finished, unless either future revisions of the specification ensure that keys are prematurely rotated upon revocation of a device, or users are informed accordingly about revocation behavior such that they understand the implications of their interactions. We discussed why edge routers, bridges and controllers hold the most power in fabrics, and that bridges as well as bridged devices especially are very interesting targets for adversaries, because all of them and the bridge share the union of permissions required by each individually bridged device and the bridge itself. Finally, we argued why we consider Matter-native OTA DFU to be secure, while we express our concerns about legacy update methods and stress out the importance of properly securing any sensitive information such as cryptographic key material during in-field upgrades to Matter. As a by-product of our discussion, we created a test catalog that is found in Appendix E and can be used by security researchers, pentesters, protocol engineers and manufacturers alike to check their implementation against a set of features which we found might either cause or prevent weaknesses.

In conclusion, we point out that Matter should be regarded as very secure and that we believe the CSA's approach on creating collaboration between a significant number of manufacturers can greatly improve overall IoT security, despite our remarks which in the end merely display that there is no such thing as perfect security. Security and privacy were taken very seriously and in most cases prioritized over usability throughout the specification, with minor and reasonable deviations. Our work is by no means complete and has yet to be confirmed or refuted by other independent researchers. Due to the high complexity of Matter and the fact that the first Matter devices were released towards the end of completing this thesis, there was no time left to create any proofs-of-concept prior to finishing this thesis. Our work is meant to provide future research with a concise and precise description of Matter's security model that is significantly shorter than the specifica-

tion. Practical research on actual devices has yet to be done, which we aim to contribute to soon. Future research should further analyze and improve Matter's security, for which we included some suggestions at the end of the discussion chapter. We submitted our work to the CSA to allow them to be the first to learn about our security considerations, and hope that at least some of our concerns are addressed and further mitigated in the future, while we hope that our perspective from an attacker point-of-view further helps them and security researchers alike understanding how to further improve IoT security and privacy.

BIBLIOGRAPHY

[1] Amr T. Abdel-Hamid, Sofiene Tahar, and El Mostapha Aboulhamid. "IP Watermarking Techniques: Survey and Comparison." In: *The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications, 2003. Proceedings.* 2003, pp. 60–65. DOI: 10.1109/IWSOC.2003.1213006.

[2] Dimitrios-Georgios Akestoridis, Vyas Sekar, and Patrick Tague. "On the Security of Thread Networks: Experimentation with OpenThread-Enabled Devices." In: *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. WiSec '22. San Antonio, TX, USA: Association for Computing Machinery, 2022, pp. 233–244. ISBN: 9781450392167. DOI: 10.1145/3507657.3528544.

[3] Fatima Zahra Amara et al. "Semantic Web and Internet of Things: Challenges, Applications and Perspectives." In: *Journal of ICT Standardization* 10.2 (May 7, 2022). DOI: 10.13052/jicts2245-800X.1029.

[4] Elena Andreeva et al. "How to Securely Release Unverified Plaintext in Authenticated Encryption." In: *Advances in Cryptology – ASIACRYPT 2014*. Ed. by Palash Sarkar and Tetsu Iwata. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 105–125. ISBN: 978-3-662-45611-8. DOI: 10.1007/978-3-662-45611-8_6.

[5] Manos Antonakakis et al. "Understanding the Mirai Botnet." In: *26th USENIX security symposium (USENIX Security 17)*. 2017, pp. 1093–1110. ISBN: 978-1-931971-40-9. URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis (visited on 08/11/2022).

[6] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. "Key Negotiation Downgrade Attacks on Bluetooth and Bluetooth Low Energy." In: *ACM Trans. Priv. Secur.* 23.3 (June 2020). ISSN: 2471-2566. DOI: 10.1145/3394497.

[7] Naama Bak et al. *Matter Security and Privacy: A Deep Dive with the Experts*. Connectivity Standards Alliance. Apr. 2022. URL: https://www.youtube.com/watch?v=Q4jhK-IBKuI (visited on 08/11/2022).

[8] Elaine Barker, Lily Chen, and Richard Davis. *Recommendation for Key-Derivation Methods in Key-Establishment Schemes*. Tech. rep. Aug. 2020. DOI: 10.6028/NIST.SP.800-56Cr2.

[9] Elaine Barker and John Kelsey. *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. Tech. rep. June 2015. DOI: 10.6028/NIST.SP.800-90Ar1.

[10] Elaine Barker et al. *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*. Tech. rep. Apr. 2018. DOI: 10.6028/NIST.SP.800-56Ar3.

[11] Simon Blake-Wilson and Alfred Menezes. "Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol." In: *Public Key Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 154–170. ISBN: 978-3-540-49162-0. DOI: 10.1007/3-540-49162-7_12.

[12] Bluetooth SIG. *Bluetooth Low Energy Core Specification 5.3*. Connectivity Standards Alliance. July 13, 2021. URL: https://www.bluetooth.com/specifications/specs/core-specification-5-3/ (visited on 01/05/2023).

[13] Sharon Boeyen et al. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. May 2008. DOI: 10.17487/RFC5280.

[14] Daniel R.L. Brown. "SEC 1: Elliptic Curve Cryptography." In: *Standards for Efficient Cryptography Group* 2 (May 2009). URL: https://www.secg.org/sec1-v2.pdf (visited on 10/15/2022).

[15] David Cash, Eike Kiltz, and Victor Shoup. *The Twin Diffie-Hellman Problem and Applications*. Cryptology ePrint Archive, Paper 2008/067. https://eprint.iacr.org/2008/067. 2008. URL: https://eprint.iacr.org/2008/067 (visited on 01/10/2023).

[16] Lily Chen et al. *Recommendations for Discrete Logarithm-Based Cryptography: Elliptic Curve Domain Parameters*. Tech. rep. Oct. 31, 2019. DOI: 10.6028/NIST.SP.800-186-draft.

[17] Shu Chen. *Matter: Bridge for Non-Matter Devices*. Espressif. Dec. 15, 2021. URL: https://blog.espressif.com/matter-bridge-for-non-matter-devices-d3b7f003a004 (visited on 09/05/2022).

[18] Shu Chen. *Matter: Distributed Compliance Ledger (DCL)*. Espressif. Mar. 25, 2022. URL: https://blog.espressif.com/matter-distributed-compliance-ledger-dcl-4013c2376e7 (visited on 09/05/2022).

[19] Stuart Cheshire and Marc Krochmal. *DNS-Based Service Discovery*. RFC 6763. Feb. 2013. DOI: 10.17487/RFC6763.

[20] Clark, D. *Estimated number of companies worldwide from 2000 to 2021, by region*. Aug. 15, 2022. URL: https://www.statista.com/statistics/1260719/global-companies-by-region/ (visited on 10/16/2022).

[21] Common Criteria Recognition Arrangement (CCRA). *Common Criteria for Information Technology Security Evaluation*. Common Criteria Recognition Arrangement (CCRA). Apr. 2017. URL: https://www.commoncriteriaportal.org/ (visited on 09/07/2022).

[22] Larry Conklin. *Threat Modeling Process*. Open Web Application Security Project (OWASP). 2022. URL: https://owasp.org/www-community/Threat_Modeling_Process (visited on 09/07/2022).

[23] Connectivity Standards Alliance. *Build With Matter*. Connectivity Standards Alliance. 2022. URL: https://csa-iot.org/all-solutions/matter/ (visited on 08/11/2022).

[24] Connectivity Standards Alliance. *Building a Safe and Secure IoT*. Connectivity Standards Alliance. Aug. 23, 2022. URL: https://csa-iot.org/certification/why-certify/ (visited on 09/05/2022).

[25] Connectivity Standards Alliance. *Certification Process*. Connectivity Standards Alliance. 2022. URL: https://csa-iot.org/certification/why-certify/ (visited on 08/11/2022).

[26] Connectivity Standards Alliance. *CSA Annual Report 2021*. Connectivity Standards Alliance. 2021. URL: https://csa-iot.org/wp-content/uploads/2022/08/21-CSA-0005_CSA-AnnualReport-ppt-wide_070822.pdf (visited on 08/11/2022).

[27] Connectivity Standards Alliance. *GitHub - Matter repository*. Connectivity Standards Alliance. 2022. URL: https://github.com/project-chip/connectedhomeip (visited on 08/11/2022).

[28] Connectivity Standards Alliance. *Matter Arrives Bringing A More Interoperable, Simple And Secure Internet Of Things to Life*. Connectivity Standards Alliance. Oct. 4, 2022. URL: https://csa-iot.org/newsroom/matter-arrives/ (visited on 10/06/2022).

[29] Connectivity Standards Alliance. *Matter Security and Privacy Fundamentals*. Tech. rep. Mar. 2022. URL: https://csa-iot.org/wp-content/uploads/2022/03/Matter_Security_and_Privacy_WP_March-2022.pdf (visited on 08/11/2022).

[30] Connectivity Standards Alliance. *Matter Specification Version 1.0*. Connectivity Standards Alliance. Oct. 4, 2022. URL: https://csa-iot.org/developer-resource/specifications-download-request/ (visited on 11/08/2022).

[31] National Vulnerability Database. *CVE-2019-9506*. 2019. URL: https://nvd.nist.gov/vuln/detail/CVE-2019-9506 (visited on 01/05/2023).

[32] Ronald B. Dekker, Fabian Otto, and Christopher Summerfield. "Curriculum learning for human compositional generalization." In: *Proceedings of the National Academy of Sciences* 119.41 (2022), e2205582119. DOI: 10.1073/pnas.2205582119.

[33] Hrishikesh Dhayagude. *Matter: Device-to-Device Automations*. Espressif. Dec. 8, 2021. URL: https://blog.espressif.com/matter-device-to-device-automations-bdbb32365350 (visited on 09/05/2022).

[34] Quang Do, Ben Martini, and Kim-Kwang Raymond Choo. "The role of the adversary model in applied security research." In: *Computers & Security* 81 (2019), pp. 156–181. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2018.12.002.

[35] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*. Tech. rep. Dec. 2001. DOI: 10.6028/NIST.SP.800-38A.

[36] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality*. Tech. rep. May 2004. DOI: 10.6028/NIST.SP.800-38C.

[37] Sascha Fahl et al. "Hey, You, Get Off of My Clipboard - On How Usability Trumps Security in Android Password Managers." In: (2013). DOI: 10.1007/978-3-642-39884-1_12.

[38] Andrea Ganassin. "Analisi del protocollo Matter ed implementazione del device type Thermostat su un prototipo (Analysis of the Matter protocol and implementation of the Thermostat device type on a prototype)." MA thesis. Universita degli Studi di Padova, Apr. 2022. URL: http://hdl.handle.net/20.500.12608/29050 (visited on 08/11/2022).

[39] Global Partners Digital. *World map of encryption laws and policies*. 2023. URL: https://www.gp-digital.org/world-map-of-encryption/ (visited on 01/21/2023).

[40] Mike Goodwin, Jon Gadsden, and Leo Reading. *OWASP Threat Dragon*. Open Web Application Security Project (OWASP). Feb. 22, 2020. URL: https://owasp.org/www-project-threat-dragon/ (visited on 09/07/2022).

[41] Google Developers. *Android 12 Behavior changes: all apps*. Jan. 23, 2023. URL: https://developer.android.com/about/versions/12/behavior-changes-all#clipboard-access-notifications (visited on 01/24/2023).

[42] Google Developers. *Privacy changes in Android 10*. Jan. 23, 2023. URL: https://developer.android.com/about/versions/10/privacy/changes#clipboard-data (visited on 01/24/2023).

[43] Greg Pearce. *Android Clipboard Security Hole Demonstration App*. 2017. URL: https://github.com/grepx/android-clipboard-security (visited on 01/24/2023).

[44] Diarmaid Harkin, Adam Molnar, and Erica Vowles. "The commodification of mobile phone surveillance: An analysis of the consumer spyware industry." In: *Crime, Media, Culture* 16.1 (2020), pp. 33–60. DOI: 10.1177/1741659018820562.

[45] Nadia Heninger et al. "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices." In: *Proceedings of the 21st USENIX Conference on Security Symposium*. Security'12. Bellevue, WA: USENIX Association, 2012, p. 35. URL: https://dl.acm.org/doi/10.5555/2362793.2362828 (visited on 09/05/2022).

[46] Infineon Technologies AG. *Can Matter finally crack the smart home?* Tech. rep. June 2022. URL: https://www.infineon.com/cms/en/product/gated-document/can-matter-finally-crack-the-smart-home-8ac78c8c821f3890018281cc3fff2d26/ (visited on 08/11/2022).

[47] Infineon Technologies AG. *The Matter Standard: Implementing Improved Security and Connectivity for the Smart Home*. Tech. rep. May 2021. URL: https://www.infineon.com/dgdl/Infineon-Matter_Connected_Things-Whitepaper-v02_00-EN.pdf?fileId=5546d46279cccfdb017a0ae13d3427c3 (visited on 08/11/2022).

[48] Guo Jiacheng. *Matter security model*. Espressif. Feb. 25, 2022. URL: https://blog.espressif.com/matter-security-model-37f806d3b0b2 (visited on 09/05/2022).

[49] Jovanovic, Bojan. *Internet of Things statistics for 2022 - Taking Things Apart*. May 13, 2022. URL: https://dataprot.net/statistics/iot-statistics/# (visited on 10/16/2022).

[50] Hugo Krawczyk. "SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols." In: *Advances in Cryptology - CRYPTO 2003*. Ed. by Dan Boneh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 400–425. ISBN: 978-3-540-45146-4. DOI: 10.1007/978-3-540-45146-4_24.

[51] Vishal Kumkar et al. "Vulnerabilities of Wireless Security protocols (WEP and WPA2)." In: *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)* 1.2 (Apr. 2012), pp. 34–38. ISSN: 2278-1323. URL: https://www.researchgate.net/publication/266005431_Vulnerabilities_of_Wireless_Security_protocols_WEP_and_WPA2 (visited on 12/07/2022).

[52] Watson Ladd and Benjamin Kaduk. *SPAKE2, a PAKE*. Internet-Draft draft-irtf-cfrg-spake2-26. Work in Progress. Internet Engineering Task Force, Feb. 2022. 17 pp. URL: https://datatracker.ietf.org/doc/draft-irtf-cfrg-spake2/26/.

[53] Stephanie L. Long. "Long Distance Bluetooth Low Energy Exploitation on a Wireless Attack Platform." MA thesis. Air Force Institute of Technology, 2021. URL: https://scholar.afit.edu/etd/5035/ (visited on 08/14/2022).

[54] Krzysztof Loska. *Introduction to Matter*. Nordic Semiconductor. Sept. 10, 2021. URL: https://www.youtube.com/watch?v=v_285vCHifw (visited on 08/11/2022).

[55] Mark Maguire. "The birth of biometric security." In: *Anthropology today* 25.2 (2009), pp. 9–14. DOI: 10.1111/j.1467-8322.2009.00654.x.

[56] Michelle Mindala-Freeman et al. *The Future of IoT is Now: Project Connected Home over IP*. Connectivity Standards Alliance. Apr. 15, 2021. URL: https://www.youtube.com/watch?v=Dqy6ASRgWmI (visited on 08/12/2022).

[57] Suvda Myagmar, Adam Lee, and William Yurcik. "Threat Modeling as a Basis for Security Requirements." In: (Aug. 2005). URL: https://www.researchgate.net/publication/228634178_Threat_Modeling_as_a_Basis_for_Security_Requirements (visited on 08/07/2022).

[58] National Institute of Standards and Technology. *Digital Signature Standard (DSS)*. Tech. rep. July 19, 2013. DOI: 10.6028/NIST.FIPS.186-4.

[59] National Institute of Standards and Technology. *Secure Hash Standard (SHS)*. Tech. rep. Aug. 4, 2015. DOI: 10.6028/NIST.FIPS.180-4.

[60] National Institute of Standards and Technology. *The Keyed-Hash Message Authentication Code (HMAC)*. Tech. rep. July 16, 2008. DOI: 10.6028/NIST.FIPS.198-1.

[61] Nordic Semiconductor. *Matter architecture*. Nordic Semiconductor. 2022. URL: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/ug_matter_overview_architecture.html (visited on 09/08/2022).

[62] Nordic Semiconductor. *Matter development model*. Nordic Semiconductor. 2022. URL: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/ug_matter_overview_dev_model.html (visited on 09/08/2022).

[63] Nordic Semiconductor. *Matter Device Attestation*. Nordic Semiconductor. 2022. URL: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/ug_matter_device_attestation.html (visited on 09/08/2022).

[64] Nordic Semiconductor. *Matter network commissioning*. Nordic Semiconductor. 2023. URL: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/ug_matter_overview_commissioning.html (visited on 01/05/2023).

[65] Nordic Semiconductor. *Matter network overview*. Nordic Semiconductor. 2022. URL: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/ug_matter_overview_network_topologies.html (visited on 09/08/2022).

[66] Andriy Panchenko and Lexi Pimenidis. "Towards Practical Attacker Classification for Risk Analysis in Anonymous Communication." In: *Communications and Multimedia Security*. Ed. by Herbert Leitold and Evangelos P. Markatos. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 240–251. ISBN: 978-3-540-47823-2. DOI: 10.1007/11909033_22.

[67] Keyur Patel and Sunil Patel. "Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges." In: *International journal of engineering science and computing* 6.5 (2016). URL: https://www.researchgate.net/publication/330425585_Internet_of_Things-IOT_Definition_Characteristics_Architecture_Enabling_Technologies_Application_Future_Challenges (visited on 08/11/2022).

[68] Stephanie K. Pell and Christopher Soghoian. "Your Secret Stingray's No Secret Anymore: The Vanishing Government Monopoly over Cell Phone Surveillance and Its Impact on National Security and Consumer Privacy." In: *Harvard Journal of Law & Technology* 28 (2014), p. 40. DOI: https://doi.org/10.31228/osf.io/67k8c.

[69] Ram Sundara Raman et al. "Investigating Large Scale HTTPS Interception in Kazakhstan." In: IMC '20. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 125–132. ISBN: 9781450381383. DOI: 10.1145/3419394.3423665.

[70] Joel Reardon et al. "50 Ways to Leak Your Data: An Exploration of Apps' Circumvention of the Android Permissions System." In: *28th USENIX Security Symposium (USENIX Security 19)*. 2019, pp. 603–620. URL: https://www.ftc.gov/system/files/documents/public_events/1415032/privacycon2019_serge_egelman.pdf (visited on 12/05/2022).

[71] Kai Ren. *Bluetooth Pairing Part 1 – Pairing Feature Exchange*. Bluetooth SIG. Mar. 29, 2016. URL: https://www.bluetooth.com/blog/bluetooth-pairing-part-1-pairing-feature-exchange (visited on 08/16/2022).

[72] Kai Ren. *Bluetooth Pairing Part 4: Bluetooth Low Energy Secure Connections – Numeric Comparison*. Bluetooth SIG. Jan. 19, 2017. URL: https://www.bluetooth.com/blog/bluetooth-pairing-part-4/ (visited on 08/16/2022).

[73] Eyal Ronen et al. "IoT Goes Nuclear: Creating a ZigBee Chain Reaction." In: *2017 IEEE Symposium on Security and Privacy (SP)*. 2017, pp. 195–212. DOI: 10.1109/SP.2017.14.

[74] Volker Roth et al. "Simple and Effective Defense against Evil Twin Access Points." In: *Proceedings of the First ACM Conference on Wireless Network Security*. WiSec '08. Alexandria, VA, USA: Association for Computing Machinery, 2008, pp. 220–235. ISBN: 9781595938145. DOI: 10.1145/1352533.1352569.

[75] David Rupprecht et al. "Breaking LTE on Layer Two." In: *IEEE Symposium on Security & Privacy (SP)*. IEEE, May 2019. DOI: 10.1109/SP.2019.00006.

[76] Mike Ryan. *Bluetooth: With Low Energy Comes Low Security*. 7th USENIX Workshop on Offensive Technologies. 2013. URL: https://www.usenix.org/system/files/conference/woot13/woot13-ryan.pdf (visited on 01/10/2023).

[77] Marc Salnikov. *Smart Home Security*. de. Bachelor Thesis. July 2022. URL: https://nbn-resolving.org/urn:nbn:de:bsz:ofb1-opus4-58451 (visited on 08/11/2022).

[78] Adam Shostack. *Threat Modeling: Designing for Security*. Wiley, Feb. 12, 2014. ISBN: 9781118810057.

[79] Victor Shoup. *Security analysis of SPAKE2+*. Cryptology ePrint Archive, Paper 2020/313. Mar. 12, 2020. URL: https://ia.cr/2020/313 (visited on 09/08/2022).

[80] Kedar Sovani. *Matter: Clusters, Attributes, Commands*. Espressif. Dec. 1, 2021. URL: https://blog.espressif.com/matter-clusters-attributes-commands-82b8ec1640a0 (visited on 09/05/2022).

[81] Kedar Sovani. *Matter: Multi-Admin, Identifiers, and Fabrics*. Espressif. Jan. 19, 2022. URL: https://blog.espressif.com/matter-multi-admin-identifiers-and-fabrics-a291371af365 (visited on 09/05/2022).

[82] Tim Taubert and Christopher A. Wood. *SPAKE2+, an Augmented PAKE (Draft Version 2)*. Internet-Draft draft-bar-cfrg-spake2plus-02. Internet Engineering Task Force, Dec. 2020. 19 pp. URL: https://datatracker.ietf.org/doc/draft-bar-cfrg-spake2plus/02/ (visited on 11/15/2022).

[83] Tim Taubert and Christopher A. Wood. *SPAKE2+, an Augmented PAKE (Draft Version 8)*. Internet-Draft draft-bar-cfrg-spake2plus-08. Internet Engineering Task Force, May 5, 2022. 30 pp. URL: https://datatracker.ietf.org/doc/draft-bar-cfrg-spake2plus/08/ (visited on 11/15/2022).

[84] Martin Thomson and Eric Rescorla. *Unknown Key-Share Attacks on Uses of TLS with the Session Description Protocol (SDP)*. RFC 8844. Jan. 18, 2021. DOI: 10.17487/RFC8844.

[85] Thormundsson, Bergur. *Smart home device household penetration in the United States in 2019 and 2021*. Mar. 17, 2022. URL: https://www.statista.com/statistics/1247351/smart-home-device-us-household-penetration/ (visited on 08/25/2022).

[86] Meltem Sönmez Turan et al. *Recommendation for Password-Based Key Derivation: Part 1: Storage Applications*. Tech. rep. Dec. 2010. DOI: 10.6028/NIST.SP.800-132.

[87] Meltem Sönmez Turan et al. *Recommendation for the Entropy Sources Used for Random Bit Generation*. Tech. rep. Jan. 2018. DOI: 10.6028/NIST.SP.800-90B.

[88] Mathy Vanhoef and Eyal Ronen. "Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd." In: *2020 IEEE Symposium on Security and Privacy (SP)*. 2020, pp. 517–533. DOI: 10.1109/SP40000.2020.00031.

[89] Wegner, Philipp. *IoT Platform Companies Landscape 2021/2022: Market consolidation has started*. Nov. 23, 2021. URL: https://iot-analytics.com/iot-platform-companies-landscape/ (visited on 01/10/2023).

[90] Di Wen, Hu Han, and Anil Jain. "Face Spoof Detection With Image Distortion Analysiss." In: *IEEE Transactions on Information Forensics and Security* 10.4 (2015), pp. 746–761. DOI: 10.1109/TIFS.2015.2400395.

[91] WhiteTimberwolf. *Block cipher mode of operation*. Wikipedia. 2013. URL: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#/media/File:CBC_encryption.svg (visited on 09/13/2022).

[92] WhiteTimberwolf. *Block cipher mode of operation*. Wikipedia. 2013. URL: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#/media/File:CTR_encryption_2.svg (visited on 09/13/2022).

[93] Doug Whiting, Russ Housley, and Niels Ferguson. *Counter with CBC-MAC (CCM)*. RFC 3610. Sept. 2003. DOI: 10.17487/RFC3610.

[94] WiGLE. *WiGLE: Wireless Network Mapping*. 2001. URL: https://www.wigle.net/ (visited on 01/10/2023).

[95] Wojciech Wodo and Lucjan Hanzlik. "Biometrics safety engineering in mobile devices." In: *2016 Future Technologies Conference (FTC)*. IEEE. 2016, pp. 941–947. DOI: 10.1109/FTC.2016.7821717.

[96] Victor Wolf et al. "Strategien durch Smart Services – Coopetition in der Smart Home-Branche." In: *Smart Services: Band 2: Geschäftsmodelle – Erlösmodelle – Kooperationsmodelle*. Ed. by Manfred Bruhn and Karsten Hadwich. Wiesbaden: Springer Fachmedien Wiesbaden, 2022, pp. 393–411. ISBN: 978-3-658-37346-7. DOI: 10.1007/978-3-658-37346-7_14.

[97] Kim Wuyts et al. *LINDDUN framework*. DistriNet KU Leuven. 2020. URL: https://www.linddun.org/linddun (visited on 09/07/2022).

[98] Xiao Zhang and Wenliang Du. "Attacks on Android Clipboard." In: *Detection of Intrusions and Malware, and Vulnerability Assessment*. Ed. by Sven Dietrich. Cham: Springer International Publishing, 2014, pp. 72–91. ISBN: 978-3-319-08509-8. DOI: 10.1007/978-3-319-08509-8_5.

A SECURITY CLAIMS

Table 1: Security Claims

ID	Claim	Fulfilled	Annotation
C.1	Does not inherit weaknesses of underlying communication technologies	Partially	Because Matter implements security on application layer, confidentiality and integrity are not dependent on underlying communication technologies as long as two genuine devices communicate with each other. However, availability can inherently not be independent of communication layers below, and DNS- and Wi-Fi-based attacks can result in impersonation, e.g.,there is a risk that commissionees pair with an evil twin.
C.2	Self-Contained: Security measures of the Core Specification sufficiently protect devices	Partially	We found security problems when considering user interaction with the UI that require additional protective measures, but we found no weaknesses that require additional security measures for all devices.
C.3	Well-tested cryptographic algorithms	Partially	AES-CCM is well-tested, but SPAKE2+ cannot be considered well-tested, as it is still in development, rarely known and we are not aware of any use-cases outside of Matter.
C.4	Strong keys	Yes	Matter uses 128-bit cryptographic keys in combination with a TRNG-seeded PRNG.
C.5	Password Authenticated Session Establishment (PASE) and Certificate Authenticated Session Establishment (CASE) for onboarding, attestation and operation	Yes	The onboarding flow and attestation requires the PASE, while operation requires the CASE. The PASE cannot be bypassed unless a valid operational private key and certificate pair are stolen. The CASE cannot be bypassed unless the attacker is able to hijack a session.
C.6	Device Attestation so that only genuine devices can join	Partially	Attestation procedure is sound and proof, but attestation verification by the commissioner is optional and can therefore be omitted.

Continued on next page

Table 1: Security Claims (Continued)

ID	Claim	Fulfilled	Annotation
C.7	Distributed Compliance Ledger (DCL)	Yes	The DCL is a core functionality of Matter.
C.8	Easy development of strong Matter implementations because reference implementations provided in a code repository	Yes	Reference implementations exist. Note that we did not perform a security analysis on them.
C.9	The user is not required to think about security	Partially	The security model is designed such that security is provided without user interaction. However, when considering bridges and secure storage of onboarding payload, the user might still required to take precautions for some devices.
C.10	Session Resumption	Yes	Both PASE and CASE offer resumption mechanisms to reduce workload.
C.11	Denial-of-Service (DoS) prevention	Partially	DoS prevention is inherently impossible, due to laws of physics. Still, Matter provides optional DoS mitigation strategies for the DCL and operation.
C.12	Sleepy End Devices (SEDs) are resilient	Yes	Apart from the minor observation of potential energy depletion attacks against SEDs that are inherently hard to mitigate, the same security model is applied to them.
C.13	Group communication is resilient	Partially	Group communication is strongly protected, but we found an underspecification in the revocation mechanism that can result in serious security and privacy issues depending on UI design and implementation-specific behavior.
C.14	Secure message counters	Yes	Matter uses a new Message Counter Synchronization Protocol (MCSP) that has yet to be tested, but seems to be secure.

Continued on next page

Table 1: Security Claims (Continued)

ID	Claim	Fulfilled	Annotation
C.15	Firmware integrity attestation: Measured boot and firmware attestation measurements	Yes	Firmware security is sufficiently addressed in the specification.
C.16	Crypto-flexibility	Partially	Matter fulfills the claim in so far, that cryptographic algorithms are abstracted such that each of them can be replaced. However, the composition itself is inflexible, and there is no sufficient explanation of how a large-scale distributed system from many different manufacturers is meant to work and interoperate, when for example the Password-Based Key Derivation Function (PBKDF) algorithm is replaced. This is critical, because the replacement can result in non-interoperability between the old and new Matter version, for example when symmetric secrets are computed. We object to the claim in so far, as that it will most likely contradict with the claim of interoperability, and that, when cryptographic algorithms are replaced, there is a high risk of upsetting costumers for these reasons.
C.17	Highest level of confidentiality and integrity used as is supported by civilian network standards to prevent eavesdropping and tampering	Yes	Although it is possible to argue how 'highest' is defined, we acknowledge that the security model of Matter highly values confidentiality and integrity, and it is currently a hard problem to design a more secure system.
C.18	All Matter devices provide an identity such that data is only shared between known Matter entities	Partially	The claim is fulfilled for Matter-to-Matter communication, however, the claim is wrong for bridged devices, which are part of the standard.
C.19	The standard and code are open, allowing peer-review	Yes	Code repository is publicly available, specifications are available per e-mail, after registration.

Continued on next page

Table 1: Security Claims (Continued)

ID	Claim	Fulfilled	Annotation
C.20	Data-minimization for all Matter interactions	Yes	This is true for any interactions observable by any of the non-communicating parties. Note that we did not check device type-specific interactions.
C.21	Data is bound to contexts	Yes	For example, use of *FixedInfo* for KDFs as described in NIST SP 800-56A [10].
C.22	Unique random Node IDs	Yes	We did not find any evidence contradicting the claim.
C.23	Session establishment with configurable privacy	N/A	The claim is ambiguous. It is unclear what the claim refers to. We guess that it might refer to device discovery over DNS-SD instead of session establishments.
C.24	Non-trackable IP addresses	No	Plain IPv6 is used, therefore a sniffer can record and attribute traffic to IP addresses. Device discovery via DNS-SD can reveal IP addresses. Connections to the DCL do not prevent tracking. The claim is ambiguous. It is unclear what the claim refers to.
C.25	Private message headers in sessions, with a separate privacy encryption key	Partially	The header is privacy encrypted. The claim is only partially fulfilled because the privacy encryption key is deterministically derived from the encryption key using a KDF. An attacker knowing the encryption key is able to derive the privacy key (but not the other way around).
C.26	Free choice of platform (including individual implementations), as long as it does not endanger ease of use and functionality	N/A	We are not able to verify or object to the claim.
C.27	Matter-ready platforms are available	Yes	For example, by Silicon Labs.

Continued on next page

Table 1: Security Claims (Continued)

ID	Claim	Fulfilled	Annotation
C.28	Secure-by-design	Yes	We acknowledge the high effort that has been put into the security model of Matter and consider Matter as truly secure-by-design, regardless of the below described weaknesses. We come to the conclusion, that Matter positively contributes to IoT security at large scale.
C.29	Mutual authentication	Partially	The claim is fulfilled for CASE, but during PASE, the commissioner is not authenticated towards the commissionee.
C.30	Multiple administrators and controllers	Yes	The feature is supported. Note that the specification is lacking a proper description of how administrators interact with each other.
C.31	Verified access controls to prevent unauthorized actions	Partially	Matter fulfills the claim by using ACLs with least-privilege principle. ACLs are fully enforced per-device for Matter-to-Matter communication. However, bridged devices share the access rights of the bridge, and thus, a bridged device A cannot be prevented from invoking actions for which only bridged device B has requested access rights.
C.32	No common or default passwords	Yes	The specification does not allow default passcodes. No other passwords are mentioned in the specification.
C.33	Keep software up-to-date	Partially	Daily updates are recommended, but it is not addressed how Matter deals with manufacturers that do not provide updates for security-critical issues.
C.34	Authenticate and validate software updates	Partially	Software updates are authenticated and validated, but consecutive failures may lead to a device installing the update, anyway, to prevent DoS of security-critical updates.
C.35	Monitor, identify and fix vulnerabilities as they are found	N/A	We are not able to verify or object to the claim.

Computational power, storage and services of devices:

Table 2: Resources: Devices

ID	Device Type
R.D.1	Node
R.D.2	Controller
R.D.3	Bridge
R.D.4	Edge Router
R.D.5	OTA Provider
R.D.6	OTA Requestor
R.D.7	Bridged Device

Other differentiations can be made between honest and dishonest Matter- and Non-Matter-certified devices, provisioned and unprovisioned devices, new and second-hand devices as well as opened and closed hardware. In the following, we provide details about certificates and keys. Singular forms of a noun refer to a single instance of the entity (e.g., 'manufacturer'), while an entire group would be explicitly marked (e.g., 'all manufacturers' or 'public')

Table 3: Resources: Certificates

ID	Name	Read Access	Created by	Signed by	Owner
R.C.1	PAA certificate	Public	CSA member	Self-signed (provisioned by CSA's TCOC)	CSA member, manufacturer
R.C.2	PAI certificate	Public	CSA member	PAA	Manufacturer
R.C.3	DAC	Device, commissioner, manufacturer	Manufacturer	PAI or PAA	Device
R.C.4	NOC	Fabric	Node (commissionee)	RCAC or ICAC private key (commissioner)	Node

Continued on next page

Table 3: Resources: Certificates (Continued)

ID	Name	Read Access	Created by	Signed by	Owner
R.C.5	RCAC	Fabric	Controller	Self-signed	Controller
R.C.5	ICAC	Fabric	Controller	RCAC	Controller

Each certificate has a corresponding public and private key. Naturally, their owner and creator is the owner of the certificate, who is also the only entity that can modify (write) a key. The read access of a public key is equivalent to read access of the corresponding certificate. read access to a private key to any of the keys is equivalent to the owner of the certificate.

Table 4: Resources: Implementation

ID	Name	Read Access	Write access	Created by	Owner
R.I.1	Device firmware	Device, manufacturer	Manufacturer	Manufacturer	Manufacturer
R.I.2	Device firmware update	Device, manufacturer	Manufacturer	Manufacturer	Manufacturer
R.I.3	Device hardware	-	Manufacturer	Manufacturer	device owner
R.I.4	Device configuration	Device, manufacturer	Device, manufacturer	Device, manufacturer	device
R.I.5	Device ACL	Device, administrator, commissioner	Administrator, commissioner	Commissioner, manufacturer	Device owner
R.I.6	Onboarding payload	Device owner, commissioner	(For dynamic passcodes: device)	Manufacturer	Device owner
R.I.7	CD	Public	TCOC	TCOC	Manufacturer

Table 5: Resources: Networking

ID	Name	Read Access	Write access	Created by	Owner
R.N.1	DCL	Public	Test Houses, CSA Certification Center, CSA members	CSA, DCL server administrators	CSA (coordinates distribution)
R.N.2	Fabric	Legitimate, commissioned receiver of a message	Commissioned nodes	Administrator	Owner of administrator device
R.N.3	Underlying networks (Wi-Fi, Thread, Ethernet)	Public or private	Public or private	Device owner or network administrator	Device owner

C SECURITY CONSIDERATIONS REGARDING BLE AND WI-FI

In the following, we comment on basic conditions that must be met to ensure that Wi-Fi and BLE are used securely. As briefly described in Section 3.3.1, BLE messages are secured using 128-bit AES-CCM end-to-end encryption, see also Section 2.3.1. The transmission range of BLE is limited, but significant increasements in range are possible using special antennas. For example, in 2021, Long demonstrated how a *skypie*-enabled drone could reach a maximum attack distance of 350 meters for BLE data collection [53]. Rose and Ramsey demonstrated in 2017 that they were able to open door locks from a quarter mile away [2] [3]. We therefore assume that BLE transmission range can limit the number of attackers committed enough to perform the attack, but is not able to fully prevent attacks.

Furthermore, we assume a BLE connection is secure if the connection establishment was performed securely. Several attacks against the pairing of BLE are known, which can greatly reduce its security. We briefly comment on these in the following. Matter does not specify the association model [72] that should be used for BLE pairing. Quote: "Note: Need to add link(s) to BLE specification" [30, Chapter 5.4.2.5]. In 2013, Ryan demonstrated a passive eavesdropping attack against BLE [76]. The vulnerability has been fixed in Bluetooth 4.2 by *LE Secure Connections Pairing*, which is based on an Elliptic Curve Diffie-Hellman (ECDH) key exchange on the NIST P-256 curve, AES-CCM for authenticated encryption and mutual authentication of the LTK. LE Secure Connections Pairing is meant to overcome legacy pairing issues and can only be used when both devices support it [12]. Since then it is possible to use the so called Federal Information Processing Standards (FIPS) mode or secure connections only mode, which disables backwards compatibility, mitigating the above described attack and other downgrade attacks. Since then, BLE uses two security modes with different security levels, of which only Mode 1, Level 4 enforces secure connections pairing [12].

During the pairing request, a maximum encryption key size is negotiated in the range of 7 to 16 bytes [71]. The Key Negotiation of Bluetooth (KNOB) attack, known as CVE-2019-9506 [31], exploits this by shortening the key down to the minimum required 7 bytes, potentially allowing to brute-force the LTK and session key. This affects BLE up to and including version 5.1. Antonioli et al. [6] demonstrated the attack in 2020 and fully compromised the security of both Bluetooth classic and BLE when they were able to decrypt any cipher text. According to them, the attack worked on any of the Bluetooth version, security modes and levels (including FIPS-mode) and device implementation details at the time. BLE-based re-pairing attacks are not expected to occur since Matter devices use BLE only during commissioning and the attacker is not able to force re-negotiation of keys. Therefore, the adversary must be able to capture the initial BLE pairing.

Matter supports the following Wi-Fi security protocols: Unencrypted, WEP, WPA-personal, WPA2-personal and WPA3-personal [30, Chapter 11.8.6.1]. Kumkar et al. [51] describe the vulner-

2 Roughly 400 meters in the metric system.

3 DEF CON 24 - Picking Bluetooth Low Energy Locks a Quarter Mile Away. https://www.youtube.com/watch?v=8h9nbMB1eTE [Last Accessed: 2022-08-17]

abilities of WEP and WPA/WPA2, which make the network vulnerable against intrusion. In 2020, Vanhoef and Ronen introduced an attack on the WPA3 dragonfly handshake [88]

TAMARIN THEORY: SPAKE2+

```
theory spake2p
begin

builtins:
  hashing, symmetric-encryption, diffie-hellman,
  bilinear-pairing

functions:
  split1/1, split2/1, merge/2, hmac/2, kdf/1,
  true/0, len/1, sub/2, add/2

equations:
  merge(split1(x), split2(x)) = x

rule passcode:
  let
    w0w1 = kdf(passcode)
    w0 = split1(w0w1)
    w1 = split2(w0w1)
  in
  [
    Fr(passcode)
  ]--[
    OnlyOnceV('passcode')
  ]->
  [
    !Prover_w0($B, w0),
    !Prover_w1($B, w1),
    !Verifier_w1($B, w1),
    !Verifier_w0($B, w0)
  ]

rule preamble_pake_values_verifier_commissionee_responder:
  let
    L = pmult(w1,'P')
  in
```

```
  [
    !Verifier_w1($B, w1),
    !Verifier_w0($B, w0)
  ]
  --[
    OnlyOnceV('preamble')
  ]->[
    !Verifier_L($B, L)
  ]

rule send_pA:
  let
    pA = add(pmult(~x, 'P'), pmult(w0, 'M'))
  in
  [
    Fr(~x),
    Fr(~session_id),
    !Prover_w0($B, w0)
  ]--[
    ProverSent_pA(~session_id, $A, $B, pA),
    OnlyOnceV('pA')
  ]->[
    Out(<~session_id, 'pA', pA>),
    !Prover_x(~session_id, $A, $B, ~x),
    !Prover_pA(~session_id, $A, $B, pA)
  ]

rule send_pB:
  let
    pB = add(pmult(~y, 'P'), pmult(w0, 'N'))
  in
  [
    Fr(~y),
    In(<~session_id, 'pA', pA>),
    !Verifier_w0($B, w0)
  ]--[
    VerifierReceived_pA(~session_id, $B, $A, pA),
    VerifierSent_pB(~session_id, $B, $A, pB), OnlyOnceV('pB')
  ]->[
    Out(<~session_id, 'pB', pB>),
    !Verifier_y(~session_id, $B, $A, ~y),
    !Verifier_pB(~session_id, $B, $A, pB),
    !Verifier_pA(~session_id, $B, $A, pA)
```

```
  ]

rule calculate_Z_prover:
  let
    Z = pmult(pmult('h', ~x), pmult(~y, 'P') )
  in
  [
    In(<~session_id, 'pB', pB>),
    !Prover_x(~session_id, $A, $B, ~x),
    !Prover_w0($B, w0),
    !Verifier_y(~session_id, $B, $A, ~y)
  ]--[
    ProverReceived_pB(~session_id, $A, $B, pB),
    OnlyOnceV('Z_prover')
  ]->[
    !Prover_Z(~session_id, $A, $B, Z),
    !Prover_pB(~session_id, $A, $B, pB),
    !Prover_y(~session_id, $A, $B, ~y)
  ]

rule calculate_Z_verifier:
  let
    Z = pmult(pmult('h', ~x), pmult(~y, 'P'))
  in
  [
    !Verifier_y(~session_id, $B, $A, ~y),
    !Verifier_pA(~session_id, $B, $A, pA),
    !Verifier_w0($B, w0),
    !Prover_x(~session_id, $A, $B, ~x)
  ]--[
    OnlyOnceV('Z_verifier')
  ]->[
    !Verifier_Z(~session_id, $B, $A, Z)
  ]

rule calculate_V_prover:
  let
    V = pmult('h', pmult(w1, pmult(~y, 'P')))
  in
  [
    !Prover_w1($B, w1),
    !Prover_w0($B, w0),
    !Prover_y(~session_id, $A, $B, ~y),
```

```
123     !Prover_pB(~session_id, $A, $B, pB)
124   ]--[
125     OnlyOnceV('V_prover')
126   ]->[
127     !Prover_V(~session_id, $A, $B, V)
128   ]
129
130 rule calculate_V_verifier:
131   let
132     V = pmult('h', pmult(w1, pmult(~y, 'P')))
133   in
134   [
135     !Verifier_y(~session_id, $B, $A, ~y),
136     !Verifier_w1($B, w1),
137     !Verifier_y(~session_id, $B, $A, ~y),
138     !Verifier_L($B, L)
139   ]--[
140     OnlyOnceV('V_verifier')
141   ]->[
142     !Verifier_V(~session_id, $B, $A, V)
143   ]
144
145 rule calculate_TT_prover:
146   let
147     TT = < 'Context', 'M', 'N', pA, pB, Z, V, w0 >
148   in
149   [
150     !Prover_pA(~session_id, $A, $B, pA),
151     !Prover_pB(~session_id, $A, $B, pB),
152     !Prover_Z(~session_id, $A, $B, Z),
153     !Prover_V(~session_id, $A, $B, V),
154     !Prover_w0($B, w0)
155   ]--[
156     DebugTT1(~session_id, $A, $B, pA, pB, Z, V, w0, TT)
157   ]->[
158     !Prover_TT(~session_id, $A, $B, TT)
159   ]
160
161 rule calculate_TT_verifier:
162   let
163     TT = < 'Context', 'M', 'N', pA, pB, Z, V, w0 >
164   in
165   [
```

```
166       !Verifier_pA(~session_id, $B, $A, pA),
167       !Verifier_pB(~session_id, $B, $A, pB),
168       !Verifier_Z(~session_id, $B, $A, Z),
169       !Verifier_V(~session_id, $B, $A, V),
170       !Verifier_w0($B, w0)
171     ]--[
172       DebugTT2(~session_id, $B, $A, pA, pB, Z, V, w0, TT)
173     ]->[
174       !Verifier_TT(~session_id, $B, $A, TT)
175     ]
176
177   rule calculate_Ka_Ke_prover:
178     let
179       hashTT = h(TT)
180       Ka = split1(hashTT)
181       Ke = split2(hashTT)
182     in
183     [
184       !Prover_TT(~session_id, $A, $B, TT)
185     ]--[
186       Secret(Ka), Secret(Ke),
187       ProverCalculated_Ka_Ke(~session_id, $A, $B, hashTT, Ka, Ke)
188     ]->[
189       !Prover_Ka(~session_id, $A, $B, Ka),
190       !Prover_Ke(~session_id, $A, $B, Ke)
191     ]
192
193   rule calculate_Ka_Ke_verifier:
194     let
195       hashTT = h(TT)
196       Ka = split1(hashTT)
197       Ke = split2(hashTT)
198     in
199     [
200       !Verifier_TT(~session_id, $B, $A, TT)
201     ]--[
202       Secret(Ka), Secret(Ke),
203       VerifierCalculated_Ka_Ke(~session_id, $B, $A, hashTT, Ka,
          Ke)
204     ]->[
205       !Verifier_Ka(~session_id, $B, $A, Ka),
206       !Verifier_Ke(~session_id, $B, $A, Ke)
207     ]
```

```
208
209 rule calculate_KcA_KcB_prover:
210   let
211     concatenatedKeys = kdf(Ka)
212     KcA = split1(concatenatedKeys)
213     KcB = split2(concatenatedKeys)
214   in
215   [
216     !Prover_Ka(~session_id, $A, $B, Ka)
217   ]--[
218     Secret(KcA), Secret(KcB), P
219     roverCalculated_KcA_KcB(~session_id, $A, $B,
          concatenatedKeys, KcA, KcB)
220   ]->[
221     !Prover_KcA(~session_id, $A, $B, KcA),
222     !Prover_KcB(~session_id, $A, $B, KcB)
223   ]
224
225 rule calculate_KcA_KcB_verifier:
226   let
227     concatenatedKeys = kdf(Ka)
228     KcA = split1(concatenatedKeys)
229     KcB = split2(concatenatedKeys)
230   in
231   [
232     !Verifier_Ka(~session_id, $B, $A, Ka)
233   ]--[
234     Secret(KcA), Secret(KcB),
235     VerifierCalculated_KcA_KcB(~session_id, $B, $A,
          concatenatedKeys, KcA, KcB)
236   ]->[
237     !Verifier_KcA(~session_id, $B, $A, KcA),
238     !Verifier_KcB(~session_id, $B, $A, KcB)
239   ]
240
241 rule calculate_cA_cB_prover:
242   let
243     cA = hmac(KcA, pB)
244     cB = hmac(KcB, pA)
245   in
246   [
247     !Prover_KcA(~session_id, $A, $B, KcA),
248     !Prover_KcB(~session_id, $A, $B, KcB),
```

```
249       !Prover_pA(~session_id, $A, $B, pA),
250       !Prover_pB(~session_id, $A, $B, pB)
251     ]--[
252       ProverSent_cA(~session_id, $A, $B, true),
253       Debug_cA_cB_Prover(~session_id, $A, $B, pA, pB, KcA, KcB,
            cA, cB )
254     ]->[
255       !Prover_cA(~session_id, $A, $B, cA),
256       !Prover_cB(~session_id, $A, $B, cB),
257       Out(<~session_id, 'cA', cA>)
258     ]
259
260   rule calculate_cA_cB_verifier:
261     let
262       cA = hmac(KcA, pB)
263       cB = hmac(KcB, pA)
264     in
265     [
266       !Verifier_KcA(~session_id, $B, $A, KcA),
267       !Verifier_KcB(~session_id, $B, $A, KcB),
268       !Verifier_pA(~session_id, $B, $A, pA),
269       !Verifier_pB(~session_id, $B, $A, pB)
270     ]--[
271       VerifierSent_cB(~session_id, $B, $A, true),
272       Debug_cA_cB_Verifier(~session_id, $B, $A, pA, pB, KcA, KcB,
            cA, cB )
273     ]->[
274       !Verifier_cA(~session_id, $B, $A, cA),
275       !Verifier_cB(~session_id, $B, $A, cB),
276       Out(<~session_id, 'cB', cB>)
277     ]
278
279   rule confirm_cB_prover:
280     let
281       sessionConfirmed = true
282     in
283     [
284       In(<~session_id, 'cB', cB>),
285       !Prover_cB(~session_id, $A, $B, cB)
286     ]--[
287       Prover_Session(~session_id, $A, $B, sessionConfirmed)
288     ]->[ ]
289
```

```
290 rule confirm_cA_verifier:
291   let
292     sessionConfirmed = true
293   in
294   [
295     In(<~session_id, 'cA', cA>),
296     !Verifier_cA(~session_id, $B, $A, cA)
297   ]--[
298     Verifier_Session(~session_id, $B, $A, sessionConfirmed)
299   ]->[ ]
300
301 rule any_cB_received_prover:
302   [
303     In(<~session_id, 'cB', cB_in>),
304     !Prover_cB(~session_id, $A, $B, cB)
305   ]--[
306     Session_cBs(~session_id, $A, $B, cB_in, cB)
307   ]->[ ]
308
309 rule any_cA_received_verifier:
310   [
311     In(<~session_id, 'cA', cA_in>),
312     !Verifier_cA(~session_id, $B, $A, cA)
313   ]--[
314     Session_cAs(~session_id, $A, $B, cA_in, cA)
315   ]->[ ]
316
317 restriction OnlyOnceV:
318 "All #i #j x.
319   OnlyOnceV(x)@#i &
320   OnlyOnceV(x)@#j ==> #i = #j"
321
322 lemma received_same_pB:
323   exists-trace
324   "Ex id pB P V #i #j.
325     VerifierSent_pB(id, V, P, pB) @ #i &
326     ProverReceived_pB(id, P, V, pB) @ #j"
327
328 lemma received_same_pB_any_session:
329   exists-trace
330     "Ex id1 id2 pB P V #i #j.
331       VerifierSent_pB(id1, V, P, pB) @ #i &
332       ProverReceived_pB(id2, P, V, pB) @ #j"
```

```
333
334 lemma received_same_pB_any_session_any_P_V:
335       exists-trace
336         "Ex id1 id2 pB P1 P2 V1 V2 #i #j.
337           VerifierSent_pB(id1, V1, P2, pB) @ #i &
338           ProverReceived_pB(id2, P1, V2, pB) @ #j"
339
340 lemma received_any_pB:
341   exists-trace
342   "Ex id pB1 pB2 P V #i #j.
343     VerifierSent_pB(id, V, P, pB1) @ #i &
344     ProverReceived_pB(id, P, V, pB2) @ #j"
345
346 lemma equal_pA:
347   exists-trace
348   "Ex A B pA pB1 pB2 V1 Z1 id TT1 V2 Z2 w0 TT2 #i #j.
349     DebugTT1(id, A, B, pA, pB1, Z1, V1, w0, TT1) @ #i &
350     DebugTT2(id, B, A, pA, pB2, Z2, V2, w0, TT2) @ #j"
351
352 lemma equal_pB:
353     exists-trace
354     "Ex A B pA1 V1 Z1 id w0 TT1 pA2 pB V2 Z2 TT2 #i #j.
355       DebugTT1(id, A, B, pA1, pB, Z1, V1, w0, TT1) @ #i &
356       DebugTT2(id, B, A, pA2, pB, Z2, V2, w0, TT2) @ #j "
357
358 lemma equal_Z_V_w0:
359   exists-trace
360   "Ex A B pA1 pB1 V Z id TT1 pA2 pB2 TT2 w0 #i #j.
361     DebugTT1(id, A, B, pA1, pB1, Z, V, w0, TT1) @ #i &
362     DebugTT2(id, B, A, pA2, pB2, Z, V, w0, TT2) @ #j"
363
364 lemma equal_TT:
365   exists-trace
366   "Ex A B pA pB V Z id w0 TT #i #j.
367     DebugTT1(id, A, B, pA, pB, Z, V, w0, TT) @ #i &
368     DebugTT2(id, B, A, pA, pB, Z, V, w0, TT) @ #j"
369
370 lemma cA_and_cB_sent:
371         exists-trace
372         "Ex P V id #i #j.
373     ProverSent_cA(id, P, V, true) @ #i
374     & VerifierSent_cB(id, V, P, true) @ #j & #j < #i"
375
```

```
376 lemma ExchangeWorks_for_A:
377         exists-trace
378         "Ex P V id t #i.
379     Prover_Session(id, P, V, t) @ #i"
380
381 lemma ExchangeWorks_for_B:
382         exists-trace
383         "Ex P V id #i.
384     Verifier_Session(id, P, V, true) @ #i"
385
386 lemma ExchangeConfirmed_ButTwoDifferentSessions:
387         exists-trace
388         "Ex P V id1 id2 #i #j.
389     Prover_Session(id1, P, V, true) @ #i
390     & Verifier_Session(id2, V, P, true) @ #j & #j < #i"
391
392 lemma ExchangeWorks:
393         exists-trace
394         "Ex P V id #i #j.
395     Prover_Session(id, P, V, true) @ #i
396     & Verifier_Session(id, V, P, true) @ #j & #j < #i"
397 end
```

TAMARIN THEORY: PAKE

```
theory matter_pake
begin

builtins:
  hashing, symmetric-encryption, diffie-hellman,
  bilinear-pairing, multiset

functions:
  split1/1, split2/1, split3/1,
  merge2/2, merge3/3, hmac/2,
  kdf/1, pbkdf/3, true/0, add/2, sub/2,
  extractI2RKey/1, extractR2IKey/1,
  extractAttestationChallenge/1

equations:
  merge2(split1(x), split2(x)) = x,
  merge3(extractI2RKey(x), extractR2IKey(x),
     extractAttestationChallenge(x)) = x,
  merge3(split1(x), split2(x), split3(x)) = x

heuristic: i

rule generate_passcode:
  let
    w0w1 = pbkdf(passcode, salt, iterations)
    w0 = split1(w0w1)
    w1 = split2(w0w1)
    pbkdf_parameters = <salt, iterations>
  in
  [
    Fr(passcode),
    Fr(salt),
    Fr(iterations)
  ]--[
    OnlyOnceV('generate_passcode'),
    Secret('passcode', passcode),
    Secret('w0', w0),
    Secret('w1', w1)
  ]->[
    Verifier_passcode($B, passcode),
    Verifier_w1($B, w1),
```

```
    Verifier_w0($B, w0),
    Verifier_PBKDFParams($B, pbkdf_parameters)
  ]

rule preamble_pake_values_verifier_commissionee_responder:
  let
    L = pmult(w1, 'P')
  in
  [
    Verifier_w1($B, w1),
    Verifier_w0($B, w0)
  ]--[
    OnlyOnceV('preamble'),
    Secret('L', L)
  ]->[
    !Verifier_PAKEValues($B, w0, w1, L)
  ]


rule scan_passcode:
  [
    Verifier_passcode($B, passcode)
  ]--[
    OnlyOnceV('scan_passcode')
  ]-> [
    !Prover_passcode($B, passcode)
  ]

rule PBKDFParamRequest:
  let
    passcodeId = '0'
    PBKDFParamRequest = <'PBKDFParamRequest', ~initiatorRandom,
        initiatorSessionId, passcodeId, hasPBKDFParameters,
       initiatorSEDParams>
      in
  [
    !Prover_passcode($B, passcode),
    Fr(~initiatorRandom),
    Fr(initiatorSessionId),
    Fr(hasPBKDFParameters),
    Fr(initiatorSEDParams),
    Fr(~sessionId)
  ]--[
```

```
82      OnlyOnceV('PBKDFParamRequest'),
83      SentPBKDFParamRequest(~sessionId, $A, $B, PBKDFParamRequest
            )
84    ]->[
85      Prover_PBKDFParamRequest(~sessionId, $A, $B,
            PBKDFParamRequest),
86      Out(<~sessionId, $A, $B, PBKDFParamRequest>)
87    ]
88
89  rule verify_PBKDFParamRequest:
90    let
91      PBKDFParamRequest = <'PBKDFParamRequest', ~initiatorRandom,
            initiatorSessionId, passcodeId, hasPBKDFParameters,
            initiatorSEDParams>
92    in
93    [
94      In(<~sessionId, $A, $B, PBKDFParamRequest>)
95    ]--[
96      OnlyOnceV('PBKDFParamRequest_Verification'),
97      VerifiedPBKDFParamRequest(~sessionId, $B, $A,
            PBKDFParamRequest)
98    ]->[
99      Verifier_PBKDFParamRequest(~sessionId, $B, $A,
            PBKDFParamRequest),
100     PBKDFParamRequest_Verified(~sessionId, $B, $A,
            PBKDFParamRequest)
101   ]
102
103 rule PBKDFParamResponse:
104   let
105     pbkdf_parameters = <salt, iterations>
106     PBKDFParamRequest = <'PBKDFParamRequest', ~initiatorRandom,
            initiatorSessionId, passcodeId, hasPBKDFParameters,
            initiatorSEDParams>
107     PBKDFParamResponse = <'PBKDFParamResponse', ~
            initiatorRandom, ~responderRandom, responderSessionId,
            pbkdf_parameters, responderSEDParams>
108   in
109   [
110     PBKDFParamRequest_Verified(~sessionId, $B, $A,
            PBKDFParamRequest),
111     Fr(~responderRandom),
112     Fr(responderSessionId),
```

```
113     Fr(responderSEDParams),
114     Verifier_PBKDFParams($B, pbkdf_parameters)
115   ]--[
116     OnlyOnceV('PBKDFParamResponse'),
117     SentPBKDFParamResponse(~sessionId, $B, $A,
          PBKDFParamResponse)
118   ]->[
119     Verifier_PBKDFParamResponse(~sessionId, $B, $A,
          PBKDFParamResponse),
120     Out(<~sessionId, $B, $A, PBKDFParamResponse>)
121   ]
122
123 rule Pake1:
124   let
125     pbkdf_parameters = <salt, iterations>
126     PBKDFParamResponse = <'PBKDFParamResponse', ~
          initiatorRandom, ~responderRandom, responderSessionId,
          pbkdf_parameters, responderSEDParams>
127     w0w1 = pbkdf(passcode, salt, iterations)
128     w0 = split1(w0w1)
129     w1 = split2(w0w1)
130     pA = add(pmult(~x,'P'), pmult(w0,'M'))
131     Pake1 = <'Pake1', pA>
132   in
133   [
134     In(<~sessionId, $B, $A, PBKDFParamResponse>),
135     Fr(~x),
136     !Prover_passcode($B, passcode)
137   ]--[
138     OnlyOnceV('Pake1'),
139     Secret('x', ~x),
140     SentPake1(~sessionId, $A, $B, Pake1),
141     DebugPake1(~sessionId, $A, $B, w0, ~x, pA, ~initiatorRandom
          , ~responderRandom, responderSessionId, pbkdf_parameters
          , responderSEDParams, PBKDFParamResponse, Pake1)
142   ]->[
143     Prover_PBKDFParamResponse(~sessionId, $A, $B,
          PBKDFParamResponse),
144     !Prover_x(~sessionId, $A, $B, ~x),
145     Prover_pA(~sessionId, $A, $B, pA),
146     Prover_Pake1(~sessionId, $A, $B, Pake1),
147     !Prover_PAKEValues($B, w0, w1),
148     Out(<~sessionId, $A, $B, Pake1>)
```

```
149   ]
150
151 rule Pake2:
152   let
153     pB = add(pmult(~y,'P'), pmult(w0,'N'))
154     Z = pmult(pmult('h', ~x), pmult(~y, 'P') )
155     V = pmult('h', pmult(w1, pmult(~y, 'P')))
156     pbkdf_parameters = <salt, iterations>
157     PBKDFParamResponse = <'PBKDFParamResponse', ~
          initiatorRandom, ~responderRandom, responderSessionId,
          pbkdf_parameters, responderSEDParams>
158     Context = h( merge3('MatterPAKEV1Commissioning',
          PBKDFParamRequest,PBKDFParamResponse) )
159     TT = <Context, 'M', 'N', pA, pB, Z, V, w0>
160     hashTT = h(TT)
161     Ka = split1(hashTT)
162     Ke = split2(hashTT)
163     concatenatedKeys = pbkdf(Ka, salt, iterations)
164     KcA = split1(concatenatedKeys)
165     KcB = split2(concatenatedKeys)
166     cA = hmac(KcA, pB)
167     cB = hmac(KcB, pA)
168     Pake1 = <'Pake1', pA>
169     Pake2 = <'Pake2', pB, cB>
170   in
171   [
172     In(<~sessionId, $A, $B, Pake1>),
173     Fr(~y),
174     !Verifier_PAKEValues($B, w0, w1, L),
175     Verifier_PBKDFParamResponse(~sessionId, $B, $A,
          PBKDFParamResponse),
176     Verifier_PBKDFParamRequest(~sessionId, $B, $A,
          PBKDFParamRequest),
177     !Prover_x(~sessionId, $A, $B, ~x)
178   ]--[
179     Secret('Verifier_Z', Z),
180     Secret('Verifier_V', V),
181     Secret('Verifier_TT', TT),
182     Secret('Verifier_Ka', Ka),
183     Secret('Verifier_Ke', Ke),
184     Secret('Verifier_KcA', KcA),
185     Secret('Verifier_KcB', KcB),
186     Secret('y', ~y),
```

```
187     OnlyOnceV('Pake2'),
188     ReceivedPake1(~sessionId, $B, $A, Pake1), SentPake2(~
           sessionId, $B, $A, Pake2), Secret(~sessionId, Ka),
           Secret(~sessionId, Ke), Secret(~sessionId, KcA), Secret
           (~sessionId, KcB),
189     DebugPake2(~sessionId, $B, $A, ~y, pA, pB, w0, L,
           PBKDFParamRequest, PBKDFParamResponse, Z, V, TT, hashTT,
            Ka, Ke, concatenatedKeys, KcA, KcB, cA, cB, Pake1,
           Pake2)
190   ]->[
191     Verifier_Pake1(~sessionId, $B, $A, Pake1),
192     Verifier_Pake2(~sessionId, $B, $A, Pake1),
193     Verifier_y(~sessionId, $B, $A, ~y),
194     Verifier_publicShares(~sessionId, $B, $A, pA, pB),
195     Verifier_confirmationTags(~sessionId, $B, $A, cA, cB),
196     Verifier_Ke(~sessionId, $B, $A, Ke),
197     Out(<~sessionId, $B, $A, Pake2>)
198   ]
199
200 rule Pake3:
201   let
202     Z = pmult(pmult('h', ~x), pmult(~y, 'P') )
203     V = pmult('h', pmult(w1, pmult(~y, 'P')))
204     PBKDFParamRequest = <'PBKDFParamRequest', ~initiatorRandom,
            initiatorSessionId, passcodeId, hasPBKDFParameters,
           initiatorSEDParams>
205     pbkdf_parameters = <salt, iterations>
206     PBKDFParamResponse = <'PBKDFParamResponse', ~
           initiatorRandom, ~responderRandom, responderSessionId,
           pbkdf_parameters, responderSEDParams>
207     Context = h( merge3('MatterPAKEV1Commissioning',
           PBKDFParamRequest,PBKDFParamResponse) )
208     TT = <Context, 'M', 'N', pA, pB, Z, V, w0>
209     hashTT = h(TT)
210     Ka = split1(hashTT)
211     Ke = split2(hashTT)
212     concatenatedKeys = pbkdf(Ka, salt, iterations)
213     KcA = split1(concatenatedKeys)
214     KcB = split2(concatenatedKeys)
215     cA = hmac(KcA, pB)
216     cB = hmac(KcB, pA)
217     Pake2 = <'Pake2', pB, cB_in>
218     Pake3 = <'Pake3', cA>
```

```
219   in
220   [
221     In(<~sessionId, $B, $A, Pake2>),
222     !Prover_PAKEValues($B, w0, w1),
223     !Prover_x(~sessionId, $A, $B, ~x),
224     Prover_pA(~sessionId, $A, $B, pA),
225     Prover_PBKDFParamRequest(~sessionId, $A, $B,
          PBKDFParamRequest),
226     Prover_PBKDFParamResponse(~sessionId, $A, $B,
          PBKDFParamResponse),
227     Verifier_y(~sessionId, $B, $A, ~y),
228   ]--[
229     OnlyOnceV('Pake3'),
230     Secret('Prover_Z', Z),
231     Secret('Prover_V', V),
232     Secret('Prover_TT', TT),
233     Secret('Prover_Ka', Ka),
234     Secret('Prover_Ke', Ke),
235     Secret('Prover_KcA', KcA),
236     Secret('Prover_KcB', KcB),
237     ReceivedPake2(~sessionId, $A, $B, Pake2), SentPake3(~
          sessionId, $A, $B, Pake3), Secret(~sessionId, Ka),
          Secret(~sessionId, Ke), Secret(~sessionId, KcA), Secret
          (~sessionId, KcB),
238     DebugPake3(~sessionId, $A, $B, pA, pB, w0, w1, ~
          initiatorRandom, initiatorSessionId, passcodeId,
          hasPBKDFParameters, initiatorSEDParams, ~responderRandom
          , responderSessionId, pbkdf_parameters,
          responderSEDParams, PBKDFParamRequest,
          PBKDFParamResponse, Z, V, TT, hashTT, Ka, Ke,
          concatenatedKeys, KcA, KcB, cA, cB, cB_in, Pake2, Pake3)
239   ]->[
240     Prover_publicShares(~sessionId, $A, $B, pA, pB),
241     Prover_confirmationTags(~sessionId, $A, $B, cA, cB),
242     Prover_cB_in(~sessionId, $A, $B, cB_in),
243     Prover_Ke(~sessionId, $A, $B, Ke),
244     Prover_Pake2(~sessionId, $A, $B, Pake2),
245     Prover_Pake3(~sessionId, $A, $B, Pake3)
246   ]
247
248 rule Pake3_verify_cB:
249   let
250     Pake3 = <'Pake3', cA>
```

```
251     cB = cB_in
252     sessionConfirmed = true
253   in
254   [
255     Prover_confirmationTags(~sessionId, $A, $B, cA, cB),
256     Prover_cB_in(~sessionId, $A, $B, cB_in),
257     Prover_Pake3(~sessionId, $A, $B, Pake3)
258   ]--[
259     OnlyOnceV('Verify_cB'),
260     Prover_Session(~sessionId, $A, $B, sessionConfirmed)
261   ]->[
262     Out(<~sessionId, $A, $B, Pake3>),
263     Prover_Pake3(~sessionId, $A, $B, Pake3)
264   ]
265
266 rule PakeFinished:
267   let
268     Pake3 = <'Pake3', cA_in>
269     cA = cA_in
270     PakeFinished = < 'SessionEstablishmentSuccess' >
271     sessionConfirmed = true
272     SEKeys_Info = 'SessionKeys'
273     I2RKey = extractI2RKey(Ke)
274     R2IKey = extractR2IKey(Ke)
275     AttestationChallenge = extractAttestationChallenge(Ke)
276     sessionEncryptionKeys = <SEKeys_Info, I2RKey, R2IKey,
          AttestationChallenge>
277   in
278   [
279     In(<~sessionId, $A, $B, Pake3>),
280     Verifier_confirmationTags(~sessionId, $B, $A, cA, cB),
281     Verifier_Ke(~sessionId, $B, $A, Ke)
282   ]--[
283     Secret('Verifier_I2RKey', I2RKey),
284     Secret('Verifier_R2IKey', R2IKey),
285     Secret('Verifier_AttestationChallenge',
          AttestationChallenge),
286     OnlyOnceV('PakeFinished'),
287     SentPakeFinished(~sessionId, $B, $A, PakeFinished),
288     Verifier_Session(~sessionId, $B, $A, sessionConfirmed),
289     Verifier_Secret_I2RKey(~sessionId, I2RKey),
290     Verifier_Secret_R2IKey(~sessionId, R2IKey)
291   ]->[
```

```
292     Verifier_Pake3(~sessionId, $B, $A, Pake3),
293     Verifier_PakeFinished(~sessionId, $B, $A, PakeFinished),
294     Verifier_calculateSessionEncryptionKeys(~sessionId,
           sessionEncryptionKeys),
295     Out(<~sessionId, $B, $A, PakeFinished>)
296   ]
297
298 rule calculateSessionEncryptionKeys_Prover:
299   let
300     Pake3 = <'Pake3', cA_in>
301     PakeFinished = < 'SessionEstablishmentSuccess' >
302     SEKeys_Info = 'SessionKeys'
303     I2RKey = extractI2RKey(Ke)
304     R2IKey = extractR2IKey(Ke)
305     AttestationChallenge = extractAttestationChallenge(Ke)
306     sessionEncryptionKeys = <SEKeys_Info, I2RKey, R2IKey,
           AttestationChallenge>
307   in
308   [
309     In(<~sessionId, $B, $A, PakeFinished>),
310     Prover_Pake3(~sessionId, $A, $B, Pake3),
311     Prover_Ke(~sessionId, $A, $B, Ke)
312   ]--[
313     Secret('Prover_I2RKey', I2RKey),
314     Secret('Prover_R2IKey', R2IKey),
315     Secret('Prover_AttestationChallenge', AttestationChallenge)
           ,
316     OnlyOnceV('CalculateSharedKeysProver'),
317     Prover_Secret_I2RKey(~sessionId, I2RKey),
318     Prover_Secret_R2IKey(~sessionId, R2IKey)
319   ]->[
320     Prover_calculateSessionEncryptionKeys(~sessionId,
           sessionEncryptionKeys)
321   ]
322
323 restriction OnlyOnceV:
324   "All #i #j x.
325     OnlyOnceV(x)@#i &
326     OnlyOnceV(x)@#j
327   ==> #i = #j "
328
329 lemma works:
330   exists-trace
```

```
331   "Ex I2RKey R2IKey id #i #j #k #l.
332     Prover_Secret_I2RKey(id, I2RKey) @ #i &
333     Verifier_Secret_I2RKey(id, I2RKey) @ #j &
334     Prover_Secret_R2IKey(id, R2IKey) @ #k &
335     Verifier_Secret_R2IKey(id, R2IKey) @ #l "
336
337 lemma attacker_does_not_know_any_secrets:
338   "All id secret #i #j.
339     Secret(id, secret) @i
340     & K(secret) @j
341   ==> F "
342
343 end
```

TAMARIN THEORY: CASE

```
theory matterSigma
begin

builtins:
  hashing, asymmetric-encryption,
  signing, diffie-hellman, xor

functions:
  hmac/2, aead_enc/4, aead_decr_verif/4,
  ecdh/2, kdf/3, crypto_verify/3

rule create_root_of_trust:
  let
    rootPublicKey = 'g'^~rootPrivateKey
    rootKeyPair = <~rootPrivateKey, rootPublicKey>
    epochKeys = <~IpkValue, ~currentEpochKey, ~nextEpochKey>
  in
  [
    Fr(~rootPrivateKey),
    Fr(fabricId),
    Fr(~IpkValue), Fr(~currentEpochKey), Fr(~nextEpochKey)
  ]--[
    RootCreated($R),
    OnlyOnceV('RootOfTrust')
  ]->[
    RootAdministrator($R, rootKeyPair, fabricId),
    RootPublicInfo($R, rootPublicKey, fabricId),
    EpochKeys(fabricId, epochKeys)
  ]

rule rotate_epoch_keys:
  let
    epochKeys = <~IpkValue, ~currentEpochKey, ~nextEpochKey>
    epochKeysNew = <~currentEpochKey, ~nextEpochKey, ~
       newEpochKey>
  in
  [
    RootAdministrator($R, rootKeyPair, fabricId),
    EpochKeys(fabricId, epochKeys),
    Fr(~newEpochKey)
  ]--[
```

```
41      OnlyOnceV('EpochKeyRotation'),
42      EpochKeysRotated(fabricId)
43    ]->[
44      RootAdministrator($R, rootKeyPair, fabricId),
45      EpochKeys(fabricId, epochKeysNew)
46    ]
47
48  rule create_node_identity:
49    let
50      epochKeys = <~IpkValue, ~currentEpochKey, ~nextEpochKey>
51      NOPublicKey = 'g'^~NOPrivateKey
52      NOKeyPair = <~NOPrivateKey, NOPublicKey>
53      NOC = <NOKeyPair>
54    in
55    [
56      RootAdministrator($R, <rootPrivateKey, rootPublicKey>,
          fabricId),
57      EpochKeys(fabricId, epochKeys),
58      Fr(~nodeId),
59      Fr(~NOPrivateKey),
60      Fr(ICAC),
61      Fr(SEDParams)
62    ]--[
63      OnlyOnceV('NodeCreated1'),
64      NodeCreated($N)
65    ]->[
66      Node($N, ~nodeId, fabricId, NOC, ICAC, ~IpkValue, SEDParams
          ),
67      RootAdministrator($R, <rootPrivateKey, rootPublicKey>,
          fabricId),
68      EpochKeys(fabricId, epochKeys)
69    ]
70
71  rule create_node_identity_2:
72    let
73      epochKeys = <~IpkValue, ~currentEpochKey, ~nextEpochKey>
74      NOPublicKey = 'g'^~NOPrivateKey
75      NOKeyPair = <~NOPrivateKey, NOPublicKey>
76      NOC = <NOKeyPair>
77    in
78    [
79      RootAdministrator($R, <rootPrivateKey, rootPublicKey>,
          fabricId),
```

```
80      EpochKeys(fabricId, epochKeys),
81      Fr(~nodeId),
82      Fr(~NOPrivateKey),
83      Fr(ICAC),
84      Fr(SEDParams)
85    ]--[
86      OnlyOnceV('NodeCreated2'),
87      NodeCreated($N)
88    ]->[
89      Node($N, ~nodeId, fabricId, NOC, ICAC, ~IpkValue, SEDParams
          ),
90      RootAdministrator($R, <rootPrivateKey, rootPublicKey>,
          fabricId),
91      EpochKeys(fabricId, epochKeys)
92    ]
93
94  rule initiator_send_sigma1:
95    let
96      initiatorEphPubKey = 'g'^~initiatorEphPrivKey
97      destinationMessage = <~initiatorRandom, rootPublicKey,
          responderFabricId, ~responderNodeId>
98      key = ~responderIpkValue
99      message = destinationMessage
100     destinationId = hmac(key, message)
101     Sigma1 = <'Sigma1', ~initiatorRandom, ~initiatorSessionId,
          destinationId, initiatorEphPubKey, initiatorSEDParams>
102   in
103   [
104     Fr(~initiatorRandom),
105     Fr(~initiatorSessionId),
106     Fr(~initiatorEphPrivKey),
107     Fr(initiatorSEDParams),
108     Fr(~sessionId),
109     RootPublicInfo($R, rootPublicKey, responderFabricId),
110     Node($Initiator, ~initiatorNodeId, initiatorFabricId,
          initiatorNOC, initiatorICAC, ~initiatorIpkValue,
          initiatorSEDParams),
111     Node($Responder, ~responderNodeId, responderFabricId,
          responderNOC, responderICAC, ~responderIpkValue,
          responderSEDParams)
112   ]--[
113     OnlyOnceV('Sigma1'),
114     Sent_Sigma1(~sessionId, $Initiator, $Responder, Sigma1),
```

```
115   ]->[
116     Out(<~sessionId, $Initiator, $Responder, Sigma1>),
117     InitiatorSigma1(<~sessionId, $Initiator, $Responder, Sigma1
          >),
118     InitiatorEphKeyPair(~sessionId, $Initiator, $Responder, ~
          initiatorEphPrivKey, initiatorEphPubKey),
119     RootPublicInfo($R, rootPublicKey, responderFabricId),
120     Node($Initiator, ~initiatorNodeId, initiatorFabricId,
          initiatorNOC, initiatorICAC, ~initiatorIpkValue,
          initiatorSEDParams),
121     Node($Responder, ~responderNodeId, responderFabricId,
          responderNOC, responderICAC, ~responderIpkValue,
          responderSEDParams)
122   ]
123
124 rule
     responder_validate_sigma1_fail_on_resumptionId_xor_initiatorResumeMIC
     :
125   let
126     Sigma1 = <'Sigma1', ~initiatorRandom, ~initiatorSessionId,
          destinationId, initiatorEphPubKey, initiatorSEDParams,
          resumptionID_or_initiatorResumeMIC>
127   in
128   [
129     In(<~sessionId, $A, $B, Sigma1>)
130   ]--[
131     Failure_on_resumptionId_xor_initiatorResumeMIC(~sessionId,
          $A, $B, Sigma1)
132   ]->[ ]
133
134 rule responder_validate_sigma1:
135   let
136     Sigma1 = <'Sigma1', ~initiatorRandom, ~initiatorSessionId,
          destinationId, initiatorEphPubKey, initiatorSEDParams>
137   in
138   [
139     In(<~sessionId, $Initiator, $Responder, Sigma1>),
140     Node($Responder, ~responderNodeId, responderFabricId,
          responderNOC, responderICAC, ~responderIpkValue,
          responderSEDParams)
141   ]--[
142     Received_Sigma1(~sessionId, $Initiator, $Responder, Sigma1)
143   ]->[
```

```
144     !Valid_Sigma1(~sessionId, $Initiator, $Responder, Sigma1),
145     !Responder_PeerSessionId(~sessionId, $Initiator, $Responder
          , ~initiatorSessionId),
146     Node($Responder, ~responderNodeId, responderFabricId,
          responderNOC, responderICAC, ~responderIpkValue,
          responderSEDParams)
147   ]
148
149 rule responder_send_sigma2:
150   let
151     Sigma1 = <'Sigma1', ~initiatorRandom, ~initiatorSessionId,
          destinationId, initiatorEphPubKey, initiatorSEDParams>
152     responderEphPubKey = 'g'^~responderEphPrivKey
153     SharedSecret = ecdh(~responderEphPrivKey,
          initiatorEphPubKey)
154     responderNOC =  <<~responderNOPrivateKey,
          responderNOPublicKey>>
155     sigma2_tbsdata = <responderNOC, responderICAC,
          responderEphPubKey, initiatorEphPubKey>
156     message = sigma2_tbsdata
157     TBSData2Signature = sign(sigma2_tbsdata, ~
          responderNOPrivateKey)
158     TBEData2 = <responderNOC, responderICAC, TBSData2Signature,
           ~ResumptionId>
159     TranscriptHash = h(Sigma1)
160     S2K_Info = 'Sigma2'
161     epochKeys = <~IpkValue, ~currentEpochKey, ~nextEpochKey>
162     salt = <~IpkValue, ~responderRandom, responderEphPubKey,
          TranscriptHash>
163     S2K = kdf(SharedSecret, salt, S2K_Info)
164     TBEData2_A = zero
165     TBEData2_Nonce = 'NCASE_Sigma2N'
166     TBEData2Encrypted = aead_enc(S2K, TBEData2, TBEData2_A,
          TBEData2_Nonce)
167     Sigma2 = <'Sigma2', ~responderRandom, ~responderSessionId,
          responderEphPubKey, TBEData2Encrypted,
          responderSEDParams>
168   in
169   [
170     EpochKeys(responderFabricId, epochKeys),
171     Node($Initiator, ~initiatorNodeId, initiatorFabricId,
          initiatorNOC, initiatorICAC, ~initiatorIpkValue,
          initiatorSEDParams),
```

```
172     Node($Responder, ~responderNodeId, responderFabricId,
           responderNOC, responderICAC, ~responderIpkValue,
           responderSEDParams),
173     !Valid_Sigma1(~sessionId, $A, $N, Sigma1),
174     Fr(~ResumptionId),
175     Fr(~responderEphPrivKey),
176     Fr(~responderRandom),
177     Fr(~responderSessionId),
178   ]--[
179     OnlyOnceV('Sigma2'),
180     Sent_Sigma2(~sessionId, $Responder, $Initiator, Sigma2)
181   ]->[
182     EpochKeys(responderFabricId, epochKeys),
183     Node($Initiator, ~initiatorNodeId, initiatorFabricId,
           initiatorNOC, initiatorICAC, ~initiatorIpkValue,
           initiatorSEDParams),
184     Node($Responder, ~responderNodeId, responderFabricId,
           responderNOC, responderICAC, ~responderIpkValue,
           responderSEDParams),
185     Out(<~sessionId, $Responder, $Initiator, Sigma1>),
186     ResponderSharedSecret(~sessionId, $Responder, $Initiator,
           SharedSecret),
187     ResponderTranscriptHash(~sessionId, $Responder, $Initiator,
            TranscriptHash),
188     ResponderEphKeyPair(~sessionId, $Responder, $Initiator, ~
           responderEphPrivKey, responderEphPubKey)
189   ]
190
191 rule initiator_validate_sigma2:
192   let
193     Sigma2 = <'Sigma2', ~responderRandom, ~responderSessionId,
           responderEphPubKey, TBEData2Encrypted,
           responderSEDParams>
194     SharedSecret = ecdh(~initiatorEphPrivKey,
           responderEphPubKey)
195     TranscriptHash = h(Sigma1)
196     S2K_Info = 'Sigma2'
197     epochKeys = <~IpkValue, ~currentEpochKey, ~nextEpochKey>
198     salt = <~IpkValue, ~responderRandom, responderEphPubKey,
           TranscriptHash>
199     S2K = kdf(SharedSecret, salt, S2K_Info)
200     TBEData2_A = zero
201     TBEData2_Nonce = 'NCASE_Sigma2N'
```

```
202     TBEData2 = aead_decr_verif(S2K, TBEData2Encrypted,
           TBEData2_A, TBEData2_Nonce)
203     TBEData2 = <responderNOC, responderICAC, TBSData2Signature,
            ~ResumptionId>
204     responderNOC =  <<~responderNOPrivateKey,
           responderNOPublicKey>>
205     sigma2_tbsdata = <responderNOC, responderICAC,
           responderEphPubKey, initiatorEphPubKey>
206     true = verify(TBSData2Signature, sigma2_tbsdata,
           responderNOPublicKey)
207   in
208   [
209     In(<~sessionId, $Responder, $Initiator, Sigma2>),
210     InitiatorSigma1(<~sessionId, $Initiator, $Responder, Sigma1
           >),
211     Node($Initiator, ~initiatorNodeId, initiatorFabricId,
           initiatorNOC, initiatorICAC, ~initiatorIpkValue,
           initiatorSEDParams),
212     Node($Responder, ~responderNodeId, responderFabricId,
           responderNOC, responderICAC, ~responderIpkValue,
           responderSEDParams),
213     EpochKeys(responderFabricId, epochKeys),
214     InitiatorEphKeyPair(~sessionId, $Initiator, $Responder, ~
           initiatorEphPrivKey, initiatorEphPubKey),
215   ]--[
216     OnlyOnceV('Validated_Sigma2'),
217   ]->[
218     Node($Initiator, ~initiatorNodeId, initiatorFabricId,
           initiatorNOC, initiatorICAC, ~initiatorIpkValue,
           initiatorSEDParams),
219     Node($Responder, ~responderNodeId, responderFabricId,
           responderNOC, responderICAC, ~responderIpkValue,
           responderSEDParams),
220     EpochKeys(responderFabricId, epochKeys),
221     InitiatorEphKeyPair(~sessionId, $Initiator, $Responder, ~
           initiatorEphPrivKey, initiatorEphPubKey),
222     Valid_Sigma2(~sessionId, $Initiator, $Responder, Sigma2),
223     InitiatorSharedSecret(~sessionId, $Initiator, $Responder,
           SharedSecret),
224     InitiatorTranscriptHash(~sessionId, $Initiator, $Responder,
            TranscriptHash)
225   ]
226
```

```
227 rule initiator_send_sigma3:
228   let
229     Sigma2 = <'Sigma2', ~responderRandom, ~responderSessionId,
          responderEphPubKey, TBEData2Encrypted,
          responderSEDParams>
230     initiatorNOC =  <<~initiatorNOPrivateKey,
          initiatorNOPublicKey>>
231     epochKeys = <~IpkValue, ~currentEpochKey, ~nextEpochKey>
232     sigma3_tbsdata = <initiatorNOC, initiatorICAC,
          initiatorEphPubKey, responderEphPubKey>
233     TBEData3Signature = sign(tbsdata, initiatorNOPrivKey)
234     TBEData3 = <initiatorNOC, initiatorICAC, TBSData3Signature>
235     S3K_Info = 'Sigma3'
236     salt = <IpkValue, TranscriptHash>
237     S3K = kdf(SharedSecret, salt, S3K_Info)
238     TBEData3_A = empty
239     TBEData3_Nonce = 'NCASE_Sigma3N'
240     TBEData3Encrypted = aead_enc(S3K, TBEData3, TBEData3_A,
          TBEData3_Nonce)
241     Sigma3 = < 'Sigma3', TBEData3Encrypted >
242   in
243   [
244     Node($Initiator, ~initiatorNodeId, initiatorFabricId,
          initiatorNOC, initiatorICAC, ~initiatorIpkValue,
          initiatorSEDParams),
245     Node($Responder, ~responderNodeId, responderFabricId,
          responderNOC, responderICAC, ~responderIpkValue,
          responderSEDParams),
246     Valid_Sigma2(~sessionId, $Initiator, $Responder, Sigma2),
247     InitiatorEphKeyPair(~sessionId, $Initiator, $Responder, ~
          initiatorEphPrivKey, initiatorEphPubKey),
248     InitiatorSharedSecret(~sessionId, $Initiator, $Responder,
          SharedSecret),
249     InitiatorTranscriptHash(~sessionId, $Initiator, $Responder,
           TranscriptHash),
250     EpochKeys(responderFabricId, epochKeys),
251   ]--[
252     OnlyOnceV('Sigma3'),
253     Sent_Sigma3(~sessionId, $Initiator, $Responder, Sigma3)
254   ]->[
255     Node($Initiator, ~initiatorNodeId, initiatorFabricId,
          initiatorNOC, initiatorICAC, ~initiatorIpkValue,
          initiatorSEDParams),
```

```
256     Node($Responder, ~responderNodeId, responderFabricId,
           responderNOC, responderICAC, ~responderIpkValue,
           responderSEDParams),
257     InitiatorEphKeyPair(~sessionId, $Initiator, $Responder, ~
           initiatorEphPrivKey, initiatorEphPubKey),
258     EpochKeys(responderFabricId, epochKeys),
259     Out(<~sessionId, $Initiator, $Responder, Sigma3>)
260   ]
261
262 rule responder_validate_sigma3:
263   let
264     Sigma1 = <'Sigma1', ~initiatorRandom, ~initiatorSessionId,
           destinationId, initiatorEphPubKey, initiatorSEDParams>
265     Sigma3 = < 'Sigma3', TBEData3Encrypted >
266     epochKeys = <~IpkValue, ~currentEpochKey, ~nextEpochKey>
267     initiatorNOC =  <<~initiatorNOPrivateKey,
           initiatorNOPublicKey>>
268     S3K_Info = 'Sigma3'
269     salt = <IpkValue, TranscriptHash>
270     S3K = kdf(SharedSecret, salt, S3K_Info)
271     TBEData3_A = empty
272     TBEData3_Nonce = 'NCASE_Sigma3N'
273     TBEData3 = aead_decr_verif(S3K, TBEData3Encrypted,
           TBEData3_A, TBEData3_Nonce) //todo aead_decr_verif/4
274     TBEData3 = <responderNOC, responderICAC, TBSData3Signature,
            ~ResumptionId>
275     sigma3_tbsdata=<initiatorNOC, initiatorICAC,
           initiatorEphPubKey, responderEphPubKey>
276   in
277   [
278     In(<~sessionId, $Initiator, $Responder, Sigma3>),
279     Node($Initiator, ~initiatorNodeId, initiatorFabricId,
           initiatorNOC, initiatorICAC, ~initiatorIpkValue,
           initiatorSEDParams),
280     Node($Responder, ~responderNodeId, responderFabricId,
           responderNOC, responderICAC, ~responderIpkValue,
           responderSEDParams),
281     ResponderTranscriptHash(~sessionId, $Responder, $Initiator,
            TranscriptHash),
282     EpochKeys(responderFabricId, epochKeys),
283     !Valid_Sigma1(~sessionId, $Initiator, $Responder, Sigma1),
284     ResponderEphKeyPair(~sessionId, $Initiator, $Responder, ~
           responderEphPrivKey, responderEphPubKey)
```

```
285   ]--[
286     OnlyOnceV('Validated_Sigma3'),
287   ]->[
288     Node($Initiator, ~initiatorNodeId, initiatorFabricId,
          initiatorNOC, initiatorICAC, ~initiatorIpkValue,
          initiatorSEDParams),
289     Node($Responder, ~responderNodeId, responderFabricId,
          responderNOC, responderICAC, ~responderIpkValue,
          responderSEDParams),
290     Valid_Sigma3(~sessionId, $Initiator, $Responder, Sigma3)
291   ]
292
293 restriction OnlyOnceV:
294   "All #i #j x.
295     OnlyOnceV(x)@#i &
296     OnlyOnceV(x)@#j
297   ==> #i = #j "
298
299 restriction Equality:
300   "All x y #i.
301     Eq(x,y) @i
302   ==> x = y "
303
304 lemma nodes_are_created_and_have_fresh_epochKeys:
305   exists-trace
306   "Ex root node #i #j.
307     RootCreated(root) @ #i &
308     NodeCreated(node) @ #j "
309
310 lemma sent_sigma1:
311   exists-trace
312   "Ex id initiator responder sigma #i.
313     Sent_Sigma1(id, initiator, responder, sigma) @ #i "
314
315 lemma sent_sigma2:
316   exists-trace
317   "Ex id initiator responder sigma #i.
318     Sent_Sigma2(id, responder, initiator, sigma) @ #i "
319 end
```

E TEST CATALOG

The following scenarios and considerations can be used for an analysis of Matter implementations:

Table 6: Test Catalog

ID	Description
TC.1	Usage of a sufficiently strong randoms (unpredictable keys, secrets, ...)
TC.2	Keys, credentials and other sensitive material are securely stored
TC.3	Side channel attacks that could reduce the security of the AEAD or any other cryptographic function
TC.4	Ability to replay (control) messages in Matter by forcing session ID reuse with trust-first policy for message counter synchronization after device reboot, especially for commands such as opening of a door
TC.5	Predictability/Entropy of passcodes
TC.6	Offline precomputation attacks, especially against PASE (SPAKE2+) and CASE
TC.7	Impact and feasibility of UKS and MitM attacks against the PASE (SPAKE2+)
TC.8	Regular checks whether the Matter specification still provides state-of-the-art security
TC.9	Ability to manipulate or reset either the Current Time, Last Known Good UTC Time, system time or epoch time of a device, or prevent a device from updating them
TC.10	Ability to force a device (temporarily or permanently) to use either of Current Time, Last Known Good UTC Time, system time or epoch time
TC.11	Ability to prevent a device's updates of the time
TC.12	Impact of time manipulation on security of a device (certificate validity, ...)
TC.13	Transport security of sensitive information such as the private keys, passcode, onboarding payload and PAKE verifier during manufacturing, shipping and retail
TC.14	Ability of an administrator to view other administrators of a node
TC.15	Ability to read-out an NFC tag storing a passcode from closed packaging
TC.16	Ability to swap the NFC tags of two devices
TC.17	Ability to inject malicious payload into a manipulated onboarding payload
TC.18	Usage of forbidden passcodes

Continued on next page

Table 6: Test Catalog (Continued)

ID	Description
TC.19	Impact and feasibility of service discovery abuses for DoS (network congestion or targeted, DNS identifier collision, ...)
TC.20	Ability to inject malicious payload into the DNS-SD *TXT* record for the DN
TC.21	Malicious introduction of faulty behavior for devices from other manufacturers, to trick users into believing the problem is with the non-malicious device (e.g., by abusing filters in commissioning apps)
TC.22	Support of MAC address randomization
TC.23	Ability to manipulate the duration of announcement periods (device discovery)
TC.24	Rotating Device Identifiers correctly meet the requirements for any Unique ID and Rotation Counter
TC.25	Ability to fake proof of possession during passcode recovery
TC.26	Transport security of recovered passcodes (both digital and analog)
TC.27	Ability to maliciously access passcode recovery, monitoring of the process
TC.28	Notification of user when a passcode is recovered
TC.29	Ability to obtain the Unique Identifier
TC.30	Additional transport channel vulnerabilities that are inherited by Matter
TC.31	Ability to spoof PH entries during custom commissioning
TC.32	Ability to bend pointers to hacked or abandoned URLs listed in the DCL
TC.33	Impact of malicious responses to discovery requests
TC.34	Reconnaissance within fabrics by abusing the Commissionable Node Discovery mechanism
TC.35	Ability and impact of Rogue Commissioner and evil twin attacks, ability of the user to perceive the attack
TC.36	Impact of a commissioner being able to see networks visible to a commissionee (security or privacy threat)
TC.37	Ability, impact and feasibility of evil twin attacks or connection of a node to malicious network
TC.38	Device prevents HTTPS downgrade attacks by implementing HTTPS only modes
TC.39	Leakage of a passcode via the *MTop* or other URL parameters, leakage of any other sensitive information via URL parameters
TC.40	Ability, feasibility, impact and detectability of the attacker to exploit ECM or BCM to add a malicious multi-admin

Continued on next page

Table 6: Test Catalog (Continued)

ID	Description
TC.41	Active notification of the user when an additional administrator is added
TC.42	Ability to revoke permissions for malicious administrators
TC.43	Inability of malicious administrators to revoke permissions for the initial and genuine administrator
TC.44	Ability to cause premature energy depletion or DoS by modification of SED parameters
TC.45	Ability to deplete an SED's energy by preventing from returning to Idle mode (e.g., send a message to the SED shortly before it returns to Idle mode)
TC.46	Ability to temporarily DoS a device by prolonging its sleep duration or extending the retransmission rate
TC.47	Ability to deplete a device's energy by abusing the UDC door bell mechanism
TC.48	Ability to abuse device discovery for impersonation of energy depletion
TC.49	Exploitation of device-specific blacklisting behavior by sender spoofing to DoS communication between two devices
TC.50	Ability, feasibility and impact of (compressed) identifier collisions in a (multi)fabric
TC.51	Robustness of fabrics against fuzzing
TC.52	Robustness against unexpected use or reserved identifiers (e.g., primordial fabric ID)
TC.53	Ability to deplete identifier space to cause DoS
TC.54	Secure implementation of rekeying
TC.55	Ability and impact of preventing session resumption to enforce CASE without resumption
TC.56	Implementations correctly check that both *ResumptionID* and *initiatorResumeMIC* are either present at the same time, or absent
TC.57	Correct and secure validation of resumption MICs
TC.58	AEAD-encrypted messages are validated prior to encryption, isolation of unverified messages that origin from a receive buffer
TC.59	Ability and impact of the replay of *Sigma1* messages
TC.60	Impact of prematurely stored session context values, ability to inject malicious code into a stored session context
TC.61	Robustness against Opcode fuzzing

Continued on next page

Table 6: Test Catalog (Continued)

ID	Description
TC.62	Ability to reduce privacy of destination identifiers or randomness by sending status report messages (*BUSY*, ...) in response to PASE- or CASE-initiating messages to enforce a restart of the session establishment with new randoms, e.g., by enticing the initiator to choose a specific random for which precalculations exist
TC.63	Prevention of trust-first policy-based replay attacks that exhaust group session counter tracking
TC.64	Ability to bypass group membership checks
TC.65	Prevention of nonce reuse during groupcast communication, including subgroups
TC.66	Ability to maliciously modify a device's ACL
TC.67	Ability to impersonate the key distribution administrator or reduce entropy of operational group keys
TC.68	Secure removal of group participants such that communication is prevented from a cryptographic perspective and in accordance with user assumptions about revocation behavior (referring to epochs and revocations)
TC.69	Secure in-field upgrade (sufficiently protected key material, ...)
TC.70	Impact of (potentially) lowered security of the in-field upgraded device
TC.71	Ability to forge certificates or the Certification Declaration
TC.72	Ability of a commissionee (joining device) to confirm DA validity of the commissioner
TC.73	Ability to cause accidental wildcards in ACL entries by subject removal
TC.74	Ability to predict the *I2RKey* or spoof or replay commissioning messages in which a commissioner configures a commissionee's ACL
TC.75	Robustness against connection failure during ACL modification
TC.76	Ability to produce mismatches between actual ACL settings, group memberships, fabrics, etc. and the displayed view
TC.77	Unambiguous user notification if an ACL can temporarily not be updated
TC.78	Determination of for how long or how often ACL updates are retransmitted
TC.79	Ability to produce inconsistent ACL entries in group management, e.g., by a combination of connection failures and reboots
TC.80	Unambiguous visualization of permissions of bridges and bridges devices, in accordance with the user's expectations
TC.81	Sufficient user information about the implications of privileges granted to a bridge or bridged device

Continued on next page

Table 6: Test Catalog (Continued)

ID	Description
TC.82	Ability, feasibility and impact of replay of (potentially overlapping) timed interactions
TC.83	Secure factory reset, reset to a secure-by-design configuration, including in-field upgraded devices
TC.84	Ability to exploit confusion of timestamps used bu events to bypass monitoring
TC.85	Sufficient allocation of event buffer space for every priority level
TC.86	Ability to exploit unconstrained lists to produce asymmetric workload at the target device
TC.87	Impact of malicious configuration by a (multi)admin, including time configuration
TC.88	User choice to allow commissioning of uncertified devices
TC.89	User choice to set a password or similarly protect the commissioning app, especially when installed on a mobile device
TC.90	Exact scope of information that is revealed and their impact when HTTPS calls are intercepted
TC.91	Secure storage of sensitive information and key material

F ATTACK GRAPHS

The attack graphs are recommended to be viewed in the PDF due to their dimensions. Each attack goal (threat) is a separate attack graph and displayed on a separate page.

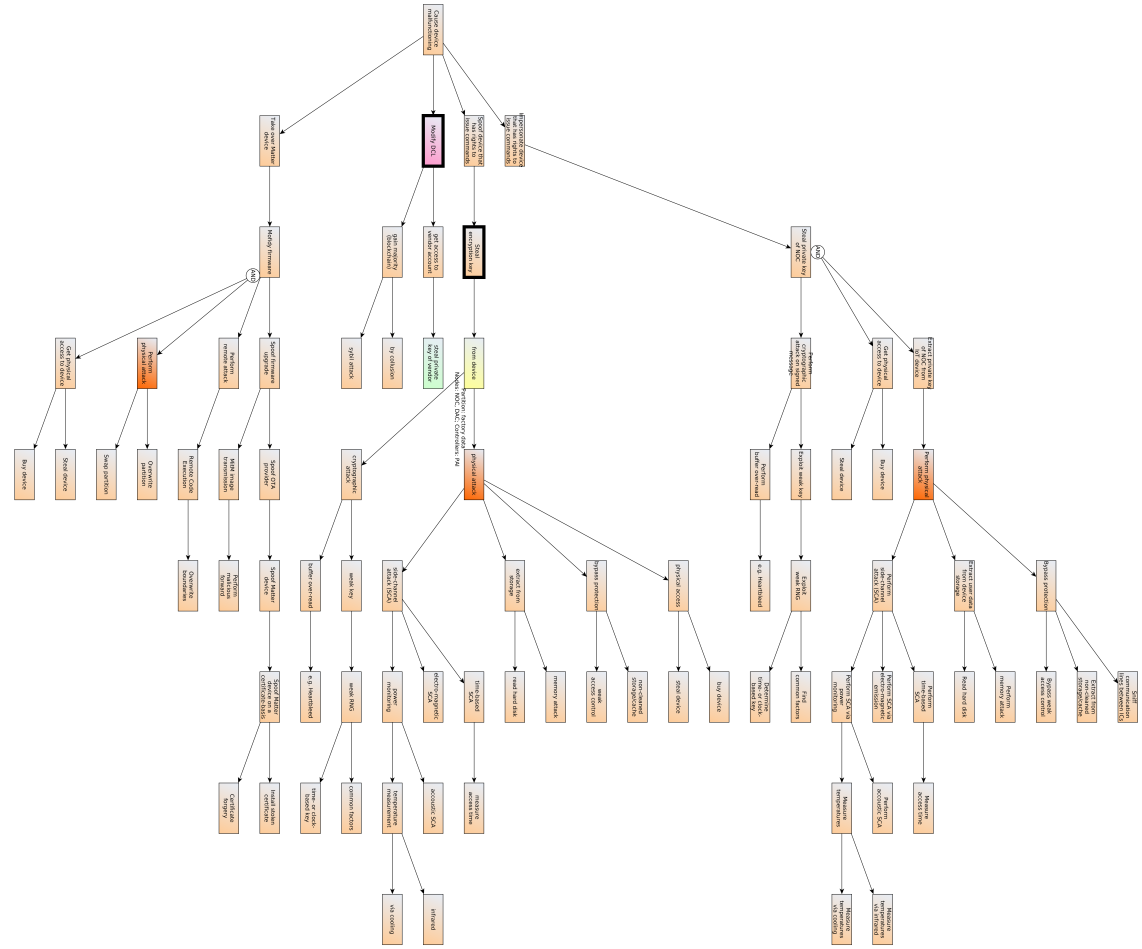


Figure 1: Attack Graph: Cause Malfunctioning

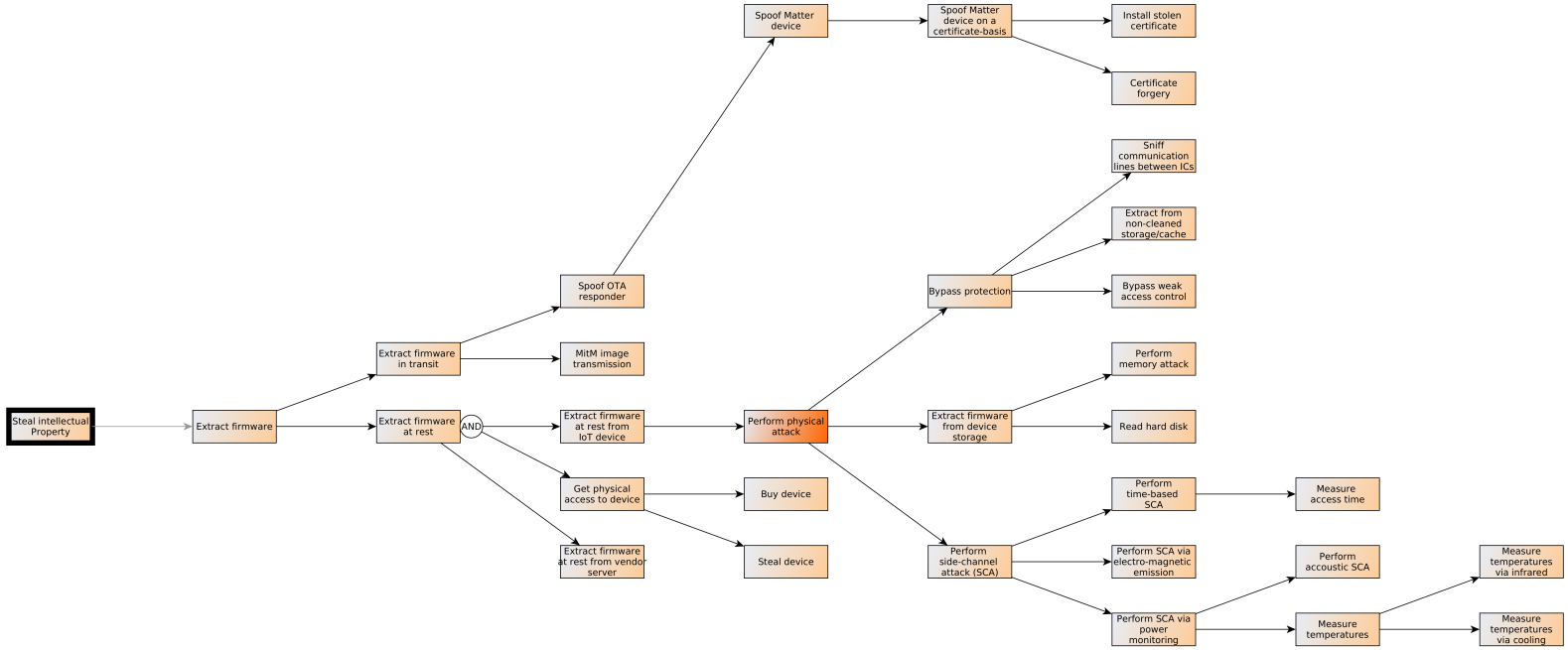


Figure 2: Attack Graph: Steal Intellectual Property

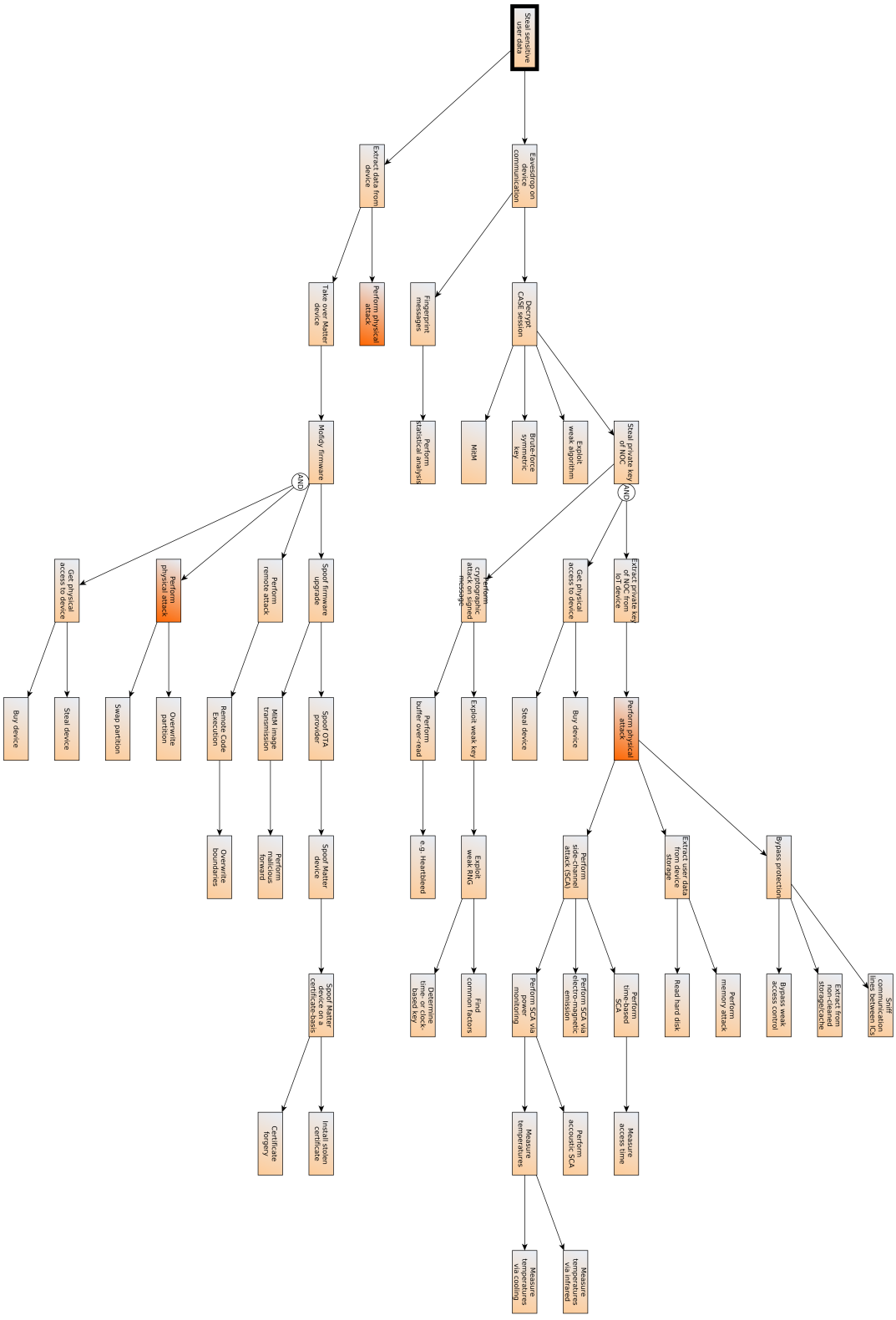


Figure 3: Attack Graph: Steal Sensitive User Data

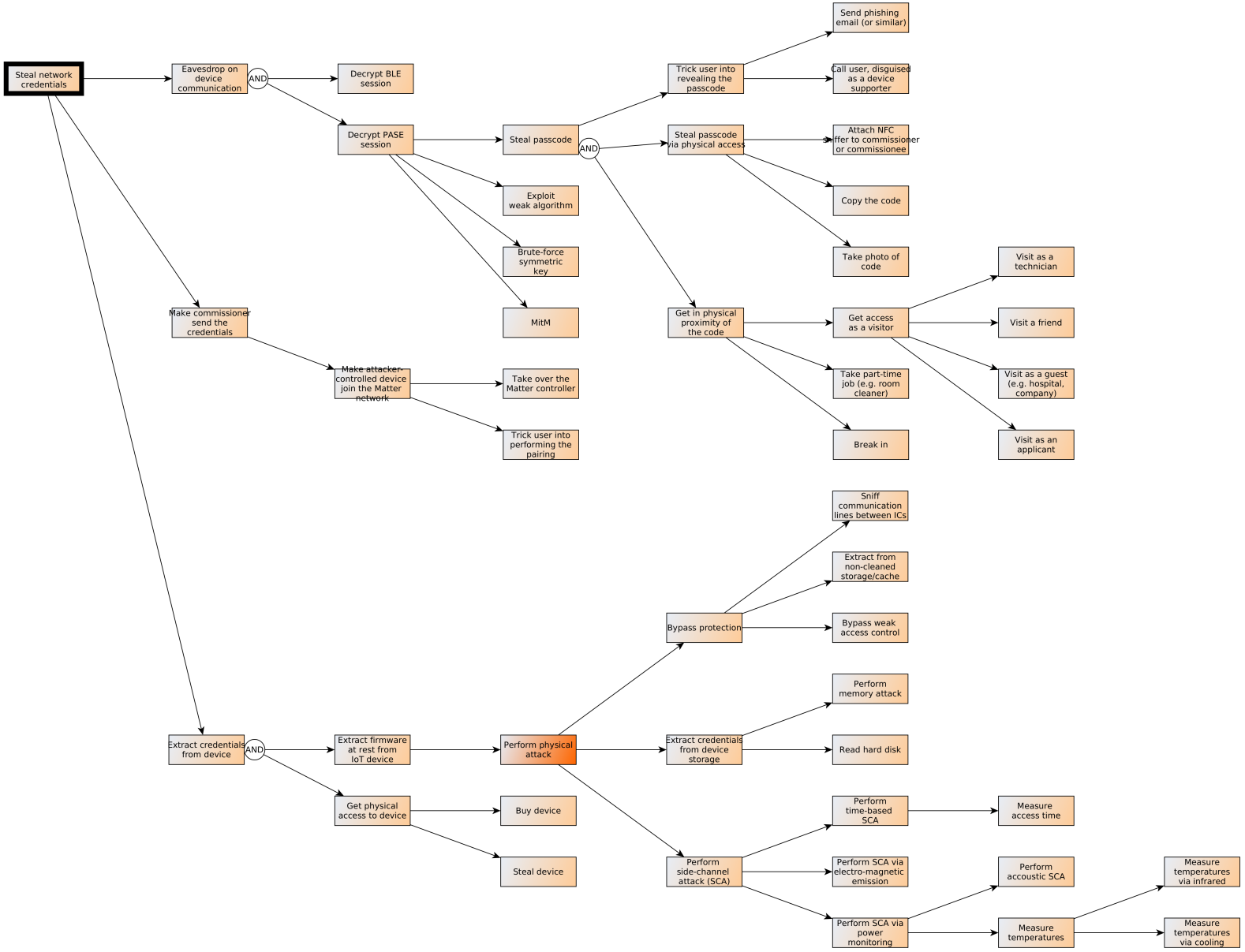


Figure 4: Attack Graph: Steal Network Credentials

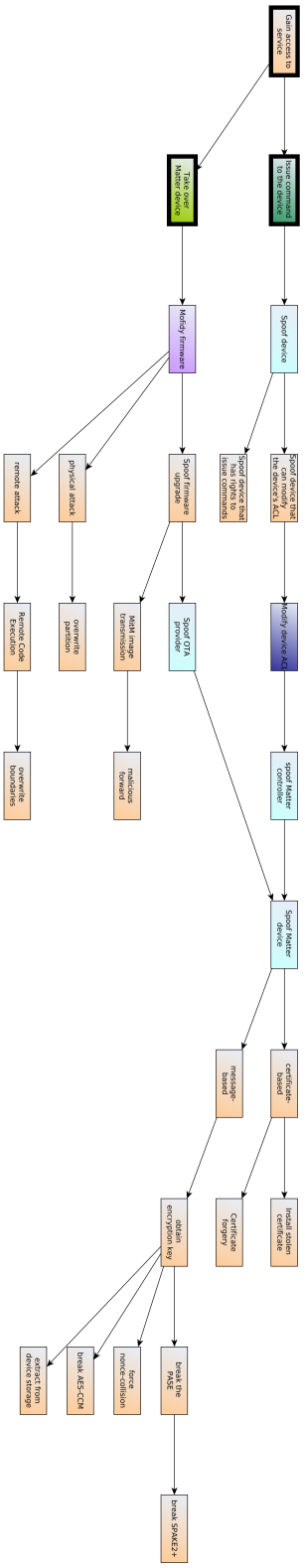


Figure 5: Attack Graph for Threat: Control Service and Use Resources

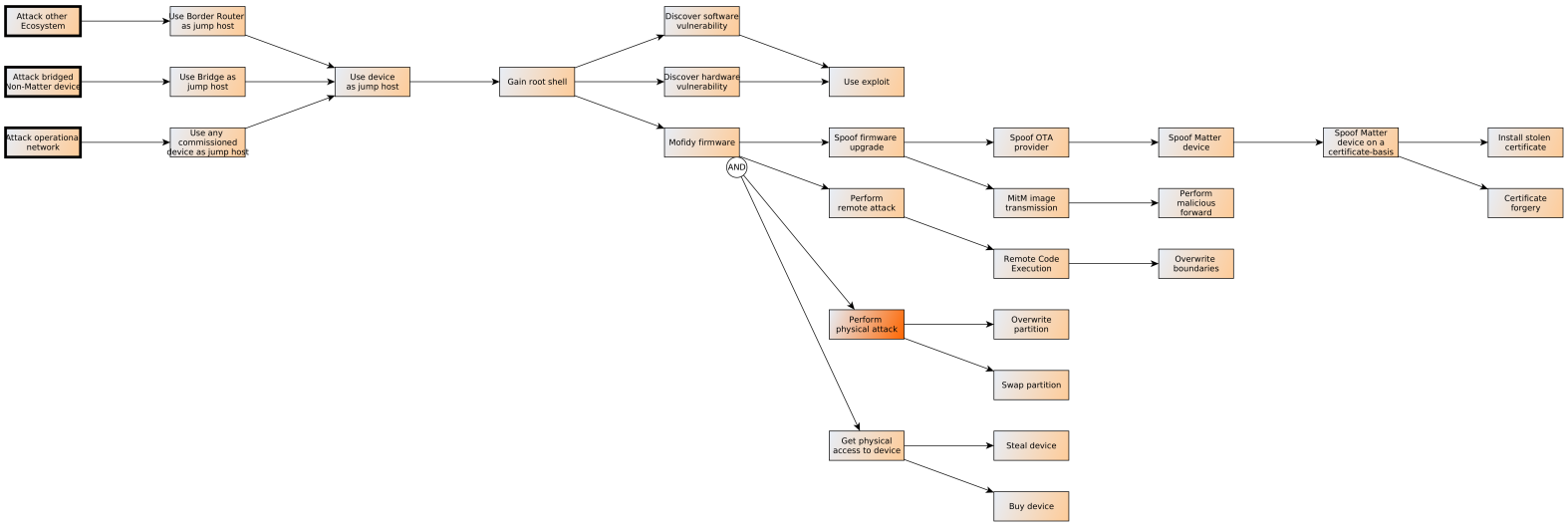


Figure 6: Attack Graph: Use Device as Jump Host

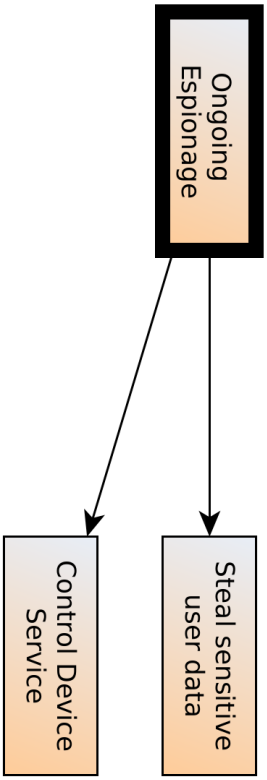


Figure 7: Attack Graph: Ongoing Espionage

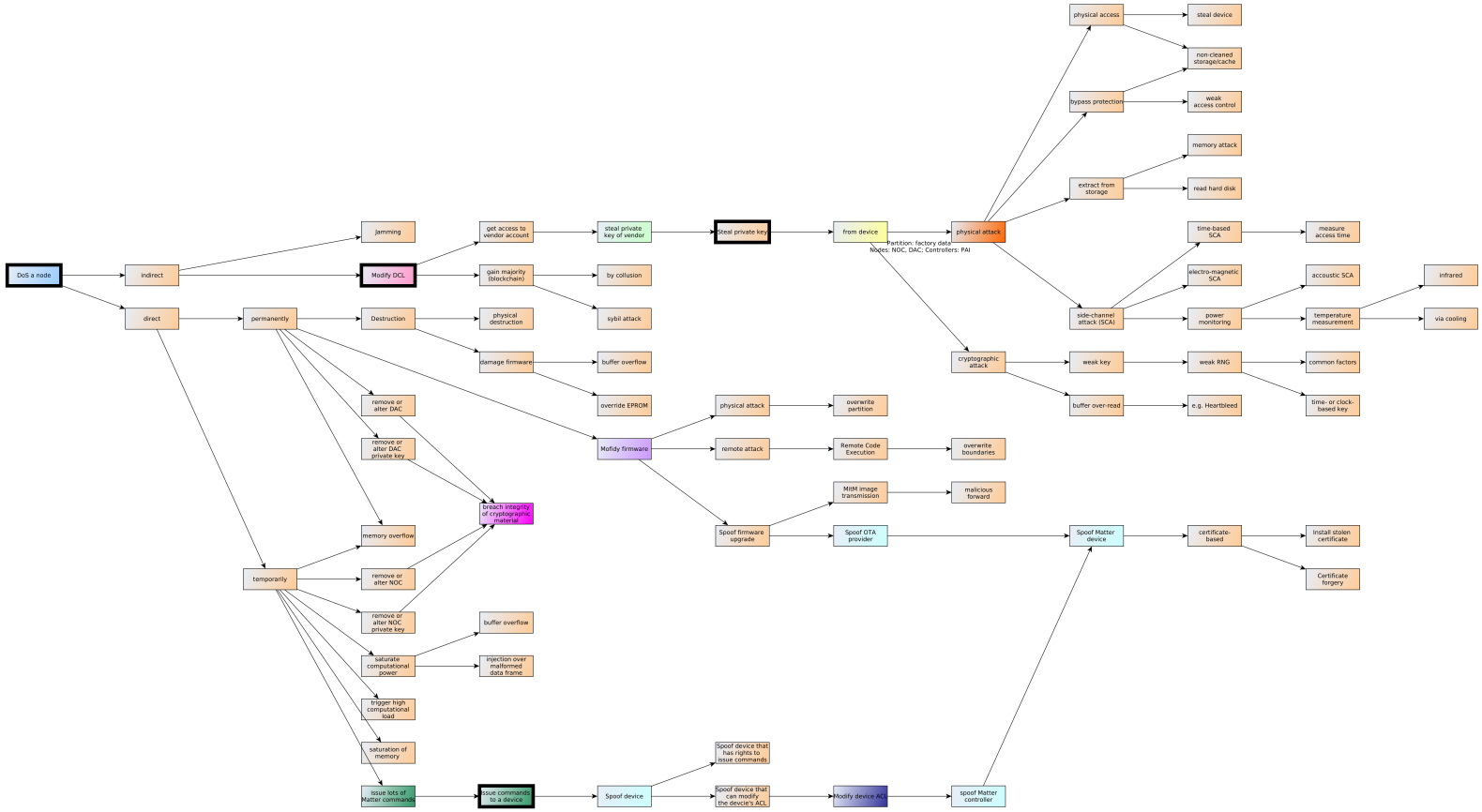


Figure 8: Attack Graph: DoS the Device

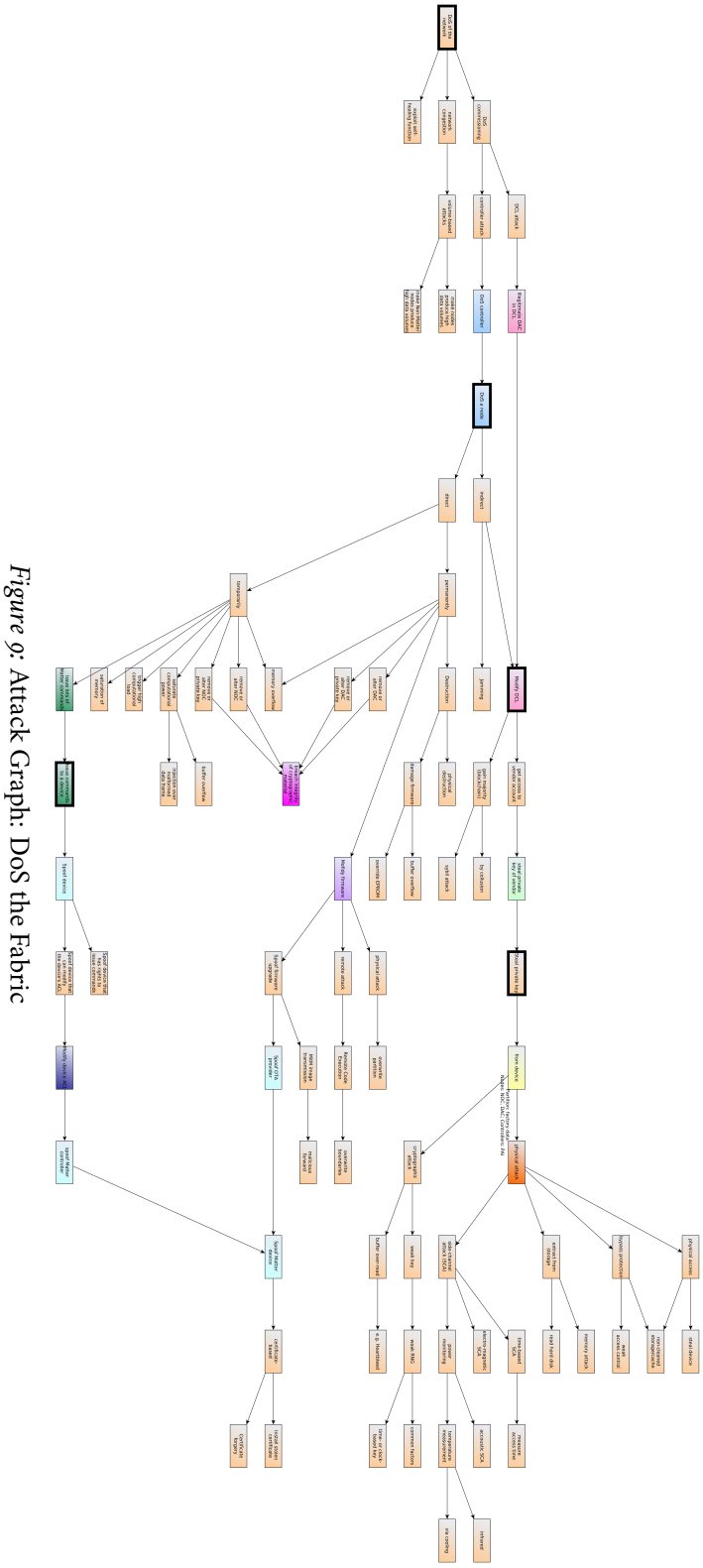


Figure 9: Attack Graph: DoS the Fabric

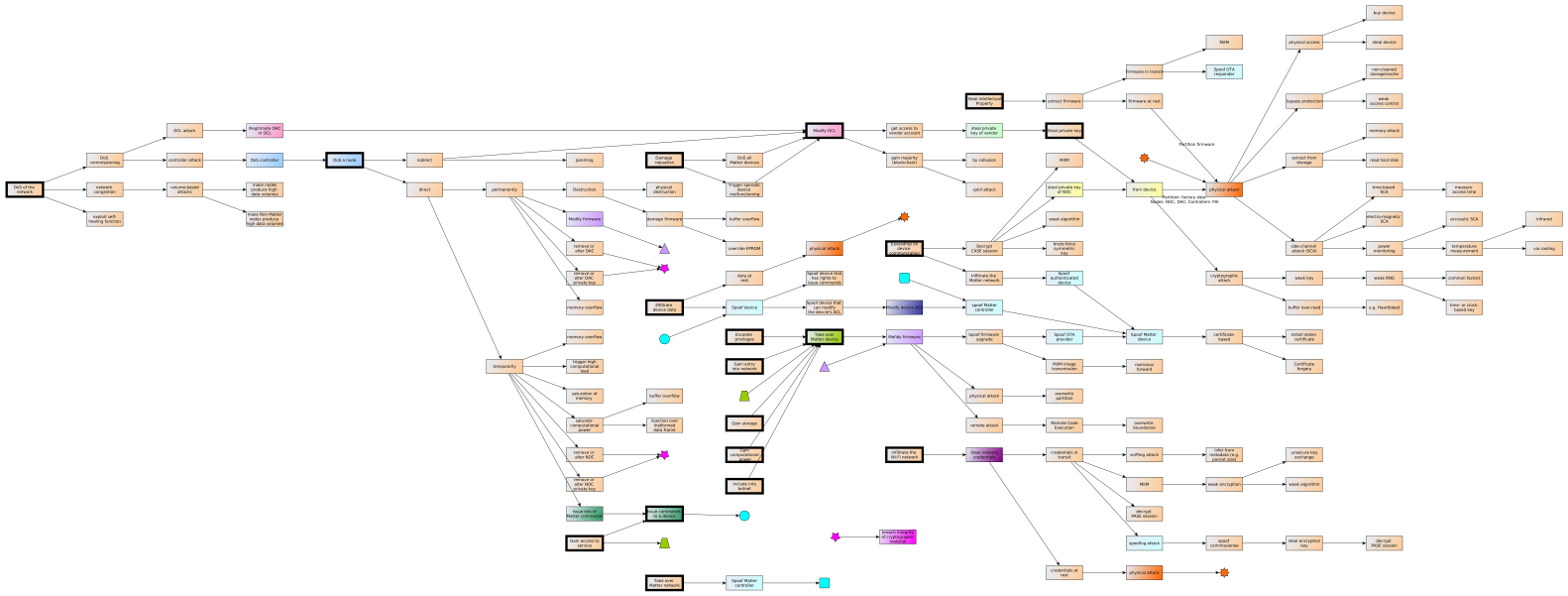


Figure 10: Aggregated Attack Graph

LIST OF FIGURES

2.1 Cipher Block Chaining mode encryption. The image is unmodified and in the public domain. Source: Wikimedia Commons [91] . . . 11
2.2 Counter mode encryption. The image is unmodified and in the public domain. Source: Wikimedia Commons [92] . . . 11
2.3 Protocol flow of SPAKE2+. Based on draft version 2 of SPAKE2+ [82] . . . 13

3.1 Matter defines the application layer within the IP stack. The image is unmodified and licensed under Apache License 2.0. Source: Matter code repository [27] . . . 17
3.2 Overview of the commissioning phases. Brown depicts unsecured communication, green is PASE-secured and blue is CASE-secured. Modified version of Nordic Semiconductor's depiction [64] . . . 18
3.3 Overview of the initial session establishment . . . 19
3.4 Example of the connectivity required between network components to allow communication between Matter devices. The image is unmodified and licensed under Apache License 2.0. Source: Matter code repository [27] . . . 22
3.5 Simplified structure of a Matter application running on the nRF Connect platform and using BLE and Thread stacks for communication purposes. The image is unmodified and licensed under Apache License 2.0. Source: Matter code repository [27] . . . 25
3.6 The protocol stack of Matter. The image is unmodified and licensed under Apache License 2.0. Source: Matter code repository [27] . . . 27
3.7 The Product Attestation Authority is stored in the Distributed Compliance Ledger, while the Product Attestation Intermediate and Device Attestation Certificate are stored in the device storage [48] . . . 29

1 Attack Graph: Cause Malfunctioning . . . 170
2 Attack Graph: Steal Intellectual Property . . . 171
3 Attack Graph: Steal Sensitive User Data . . . 172
4 Attack Graph: Steal Network Credentials . . . 173
5 Attack Graph for Threat: Control Service and Use Resources . . . 174
6 Attack Graph: Use Device as Jump Host . . . 175
7 Attack Graph: Ongoing Espionage . . . 176
8 Attack Graph: DoS the Device . . . 177
9 Attack Graph: DoS the Fabric . . . 178
10 Aggregated Attack Graph . . . 179

LIST OF TABLES

4.1 External Dependencies 35
4.2 List of what systems require protection 36
4.3 Trust Levels 37
4.4 Assets (the attacker is interested in) 38
4.5 Access Points 40
4.6 Entry Points 41
4.7 Exit Points 42
4.8 Threat Scenarios 46
4.9 Security (S) and Privacy (P) Requirements 47

1 Security Claims 121

2 Resources: Devices 127
3 Resources: Certificates 127
4 Resources: Implementation 128
5 Resources: Networking 129

6 Test Catalog 163