HOCHSCHULE
MITTWEIDA
University of Applied Sciences

# MASTERARBEIT

Frau
**Emelie Charlotte Hanske**, B.Sc.

## Untersuchung der Netzwerk-, Cloud- und App-Daten eines Yale Smart Schlosses im forensischen Kontext

Mittweida, November 2025

Fakultät **Angewandte Computer- und Biowissenschaften**

# MASTERARBEIT

# Untersuchung der Netzwerk-, Cloud- und App-Daten eines Yale Smart Schlosses im forensischen Kontext

Autorin:
**Emelie Charlotte Hanske**

Studiengang:
Cybercrime/Cybersecurity

Seminargruppe:
CY23wC-M

Erstprüfer:
Prof. Dr. rer. nat. Dirk Labudde

Zweitprüfer:
Felix Fischer, M.Sc.

Einreichung:
Mittweida, 09.11.2025

Verteidigung/Bewertung:
Mittweida, 2025

Faculty of **Applied Computer Sciences and Biosciences**

# MASTER THESIS

# Examining the network, cloud and app data of a Yale Smart Lock in a forensic context

Author:
**Emelie Charlotte Hanske**

Course of Study:
Cybercrime/Cybersecurity

Seminar Group:
CY23wC-M

First Examiner:
Prof. Dr. rer. nat. Dirk Labudde

Second Examiner:
Felix Fischer, M.Sc.

Submission:
Mittweida, 09.11.2025

Defense/Evaluation:
Mittweida, 2025

## Bibliografische Beschreibung

Hanske, Emelie Charlotte:
Untersuchung der Netzwerk-, Cloud- und App-Daten eines Yale Smart Schlosses im forensischen Kontext. – 2025. – 71 S.
Mittweida, Hochschule Mittweida – University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften, Masterarbeit, 2025.

## Gendervermerk

In dieser Arbeit wird aus Gründen der besseren Lesbarkeit das generische Maskulinum verwendet. Weibliche und anderweitige Geschlechteridentitäten werden dabei ausdrücklich mitgemeint, soweit es für die Aussage erforderlich ist.

## Nutzung von Künstlicher Intelligenz

Zur Unterstützung bei der Erstellung dieser Arbeit wurde das Sprachmodell ChatGPT (Version GPT-5, OpenAI, 2025) verwendet. Es diente ausschließlich der Klärung von Formatierungsfragen in LaTeX. Für eine verbesserte Ausdrucksweise/sprachliche Überarbeitung, zur Korrektur von Grammatik und Rechtschreibung wurde DeepL Write verwendet. Sämtliche Inhalte, Analysen und Schlussfolgerungen wurden eigenständig erarbeitet.

## Referat

Die vorliegende Arbeit untersucht forensisch relevante digitale Spuren, die im Zusammenhang mit der Nutzung eines Internet of Things (IoT)-Geräts am Beispiel des Yale Linus® Smart Locks und der zugehörigen Yale Home App entstehen. Ziel der Untersuchung ist es, zu ermitteln, welche Daten während der Nutzung des Geräts und der App generiert werden, in welcher Form diese vorliegen und an welchen Orten sie gespeichert werden. Darüber hinaus wird analysiert, welche Rückschlüsse auf das Nutzerverhalten möglich sind und ob sicherheitsrelevante Schwachstellen bestehen. Die Arbeit orientiert sich an zwei Forschungsfragen: (1) Wie lassen sich aus den während der Nutzung eines Yale Linus® Smart Locks erzeugten Netzwerk-, Cloud- und App-Daten digitale Spuren identifizieren, die forensisch relevant sind, und welchen Beitrag leisten diese zur Untersuchung sicherheitsrelevanter Vorfälle? (2) Welche Informationen lassen sich aus der zugehörigen App, über die die Aktionen gesteuert werden, extrahieren? Zur Beantwortung dieser Forschungsfragen wurden verschiedene Methodiken eingesetzt: das Mitschneiden des Netzwerkverkehrs über das Bluetooth-Host Controller Interface (HCI)-Protokoll und dessen Auswertung mit Wireshark, die Analyse der App mittels Android Debug Bridge (ADB), die Untersuchung der Cloud-Daten sowie der Vergleich des App-Protokolls mit einem eigens erstellten Protokoll. Die Ergebnisse zeigen, dass insbesondere die app-spezifischen Daten wesentliche Informationen über Nutzer- und Gerätedaten liefern. In Kombination mit den App-Protokollen und den Cloud-Daten ermöglichen sie zudem eine Rekonstruktion relevanter Ereignisabläufe.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Abkürzungsverzeichnis

**ACK** . . . . . Acknowledgement

**ADB** . . . . . Android Debug Bridge

**AES** . . . . . Advanced Encryption Standard

**App** . . . . . Anwendungssoftware

**ATT** . . . . . Attribute Protocol

**BLE** . . . . . Bluetooth Low Energy

**CCCD** . . . . . Client Characteristic Configuration Descriptor

**CCM** . . . . . Counter-Modus mit Cipher Block Chaining Message Authentication Code

**CSV** . . . . . Comma Separated Values

**DHKey** . . . . . Diffie-Hellman Key

**EAN** . . . . . European Article Number

**ECDH** . . . . . Elliptic Curve Diffie-Hellman

**GAP** . . . . . Generic Access Profile

**GATT** . . . . . Generic Attribute Profile

**GPS** . . . . . Global Positioning System

**HCI** . . . . . Host Controller Interface

**IANA** . . . . . Internet Assigned Numbers Authority

**ID** . . . . . Identifikationsnummer

**IIoT** . . . . . Industrial Internet of Things

**Inc.** . . . . . Incorporated

**iOS** . . . . . iPhone Operating System

**IoT** . . . . . Internet of Things

**IP** . . . . . Internet Protocol

**ISM** . . . . . Industrial, Scientific and Medical

**IT** . . . . . Informationstechnik

**JSON** . . . . . JavaScript Object Notation

**L2CAP** . . . . . Logical Link Control and Adaptation Protocol

**LAN** . . . . . Local Area Network

**LE** . . . . . Low Energy

**LL** ..... Link Layer

**LTK** ..... Long Term Key

**MAC** ..... Media Access Control

**NFC** ..... Near Field Communication

**NIST** ..... National Institute of Standards and Technology

**OOB** ..... Out of Band

**PCAP** ..... Packet Capture

**PHY** ..... Physical Layer

**SIG** ..... Special Interest Group

**SKU** ..... Stock Keeping Unit

**SMP** ..... Security Manager Protocol

**SQL** ..... Structured Query Language

**SSP** ..... Secure Simple Pairing

**SYN** ..... Synchornize

**TCP** ..... Transmission Control Protocol

**TK** ..... Temporary Key

**TLS** ..... Transport Layer Security

**USB** ..... Universal Serial Bus

**UTC** ..... Universal Time Coordinated

**UUID** ..... Universally Unique Identifier

**WAN** ..... Wide Area Network

**Wi-Fi** ..... Wireless Fidelity

**WLAN** ..... Wireless Local Area Network

**XML** ..... Extensible Markup Language

# 1 Einleitung

Der Begriff Internet of Things (IoT), zu dem auch Konzepte wie Industrie 4.0 zählen, beschreibt im Kern eine technologische Entwicklung, die eine umfassende Vernetzung physischer und digitaler Objekte innerhalb einer weltweiten Informations- und Kommunikationsumgebung ermöglicht [1]. Das stellt den zentralen Treiber des digitalen Wandels dar, der nicht nur die Industrie, sondern zunehmend auch den privaten Alltag prägt [2].

Die Umsetzung von IoT-Projekten erweist sich dabei in der Regel als komplex, da zahlreiche Informationstechnik (IT)-Bereiche aufeinander abgestimmt werden müssen [3]. Technologisch stützt sich das IoT auf Schlüsselbereiche wie Big Data, Cloud Computing und Künstliche Intelligenz [3]. Charakteristisch ist die enge Verzahnung von der Erfassung, Übertragung und Auswertung von Daten, die für eine effiziente und skalierbare Nutzung der Systeme unerlässlich ist [2].

In den vergangenen Jahren hat das IoT insgesamt deutlich an Bedeutung gewonnen. Durch seine grundlegende Vernetzungsfunktion — sei es zwischen Produkten, Dienstleistungen, Maschinen oder Sensoren — eröffnet es ein breites Anwendungsspektrum, das in den unterschiedlichsten Branchen genutzt werden kann. Die Einsatzmöglichkeiten reichen dabei von Smart Home und Smart Factory über Smart Health und Smart Mobility bis hin zu ganzen Smart Cities [3].

Ein besonders dynamischer Anwendungsbereich zeigt sich im häuslichen Umfeld. Unter dem Begriff Smart Home werden IoT-Technologien eingesetzt, um alltägliche Geräte und Systeme zu vernetzen. Dazu zählen beispielsweise Kühlschränke, Klimaanlagen, Beleuchtung, Heizungen, Türschlösser oder Sicherheitskameras, die sich mithilfe digitaler Schnittstellen überwachen, steuern und automatisieren lassen. [4]

## 1.1 Motivation

Ausgehend von dieser grundlegenden Bedeutung des Smart Homes zeigt eine Umfrage des Cybersicherheitsmonitors, einem Zusammenschluss der Polizeilichen Kriminalprävention der Länder und des Bundes sowie dem Bundesamt für Sicherheit in der Informationstechnik, aus dem Jahr 2024, dass 75 % der 3 047 Befragten bereits ein Smart Home-Gerät nutzen. Die Bandbreite der Anwendungen ist dabei groß: Sie reicht von häufig genutzten smarten Fernsehgeräten bis hin zu intelligentem Spielzeug, das in vielen deutschen Privathaushalten eingesetzt wird. [5]

Die Studie von Dr. Bernhard Rohleder aus demselben Jahr verdeutlicht zudem, dass für viele Nutzer die Sicherheit des eigenen Zuhauses eine zentrale Motivation darstellt. So gaben 69 % der 1 193 Befragten an, ihr Haus oder ihre Wohnung mithilfe von Smart Home-Technologien sicherer machen zu wollen. Darüber hinaus zeigt sich, dass 90 % der Teilnehmenden ihre Smart Home-Geräte über das Smartphone steuern. [6]

Vor diesem Hintergrund überrascht es nicht, dass rund 3 % der Befragten auf smarte Tür- und Fensterschlösser setzen [5]. Solche Geräte wie das Yale Linus® Smart Lock lassen sich bequem über die Anwendungssoftware (App) bedienen und eröffnen neue Möglichkeiten für einen flexiblen und zugleich sicheren Zugang zum eigenen Zuhause. Neben der komfortablen Steuerung über mobile Endgeräte zeigt sich dabei jedoch auch, dass die Erwartungen der Nutzer in hohem Maße mit Fragen der IT-Sicherheit und des Datenschutzes verknüpft sind [6]. Aus der Smart Home-Umfrage aus dem Jahr 2024 geht hervor, dass viele Konsumenten besonderen Wert auf eine verlässliche Absicherung gegenüber Hackerangriffen legen, ein unabhängiges Gütesiegel oder Zertifikat als Orientierungshilfe fordern und auch die Speicherung von Daten innerhalb der Europäischen Union als wichtigen Aspekt ansehen [6].

Die zunehmende Verbreitung von Smart Home-Technologien zeigt, dass diese Systeme mittlerweile einen festen Bestandteil des Alltags bilden und damit ein relevantes Untersuchungsfeld darstellen. Gleichzeitig legen viele Nutzer großen Wert auf die Sicherheit ihrer Geräte und des Gesamtsystems, wie aktuelle Umfragen zur Nutzung von Smart Home-Anwendungen belegen [5], [6]. Vor diesem Hintergrund stellt sich die Frage, wie Smart Home-Geräte, insbesondere das Yale Linus® Smart Lock, technisch funktionieren, welche digitalen Daten sie erfassen und speichern sowie welche potenziellen Risiken sich daraus ergeben können. Die Untersuchung dieser Fragestellungen ist sowohl aus wissenschaftlicher Perspektive als auch im Hinblick auf praktische Aspekte wie Datenschutz, IT-Sicherheit und digitale Spurensicherung von hoher Relevanz. Sie bilden die Grundlage für die folgende Untersuchung, die sich der forensischen Analyse des Yale Linus® Smart Lock widmet.

## 1.2 Problemstellung und Relevanz

Wie bereits angedeutet, stellt die forensische Untersuchung von IoT-Geräten die Fachwelt vor besondere Herausforderungen. Obwohl vernetzte Alltagsgegenstände zunehmend an Bedeutung gewinnen und in immer mehr Lebensbereiche integriert werden, existiert bislang kein einheitliches Verfahren oder etabliertes Werkzeug, das auf breiter Basis für die Analyse herangezogen werden kann. Ermittler stehen daher vor der anspruchsvollen Aufgabe, geeignete Methoden und Werkzeuge zu identifizieren, um relevante digitale Spuren erfassen zu können. Hinzu kommt, dass die durch IoT-Umgebungen erzeugte Datenmenge potenziell unbegrenzt ist, wodurch die Anwendung klassischer digitalforensischer Ansätze weltweit erheblich erschwert wird. [7]

Für das Yale Linus® Smart Lock ergibt sich eine besondere Relevanz, da dieses bislang nicht systematisch aus forensischer Perspektive untersucht wurde. Insbesondere ist bislang unklar, welche digitalen Artefakte gesichert und analysiert werden können. Eine forensische Untersuchung dieses konkreten Modells kann daher einen wichtigen Beitrag zum besseren Verständnis der digitalen Spurensicherung im Kontext von Smart Home-Geräten leisten und eine bestehende Forschungslücke schließen.

## 1.3 Zielsetzung und Forschungsfrage

Das Ziel dieser Arbeit besteht darin, Netzwerk-, Cloud- und App-Daten zu untersuchen, die im Zusammenhang mit der Nutzung des Yale Linus® Smart Lock entstehen und einen Vergleich zweier Ereignisprotokolle vorzunehmen. Dabei soll sowohl deren forensische Bedeutung erkannt als auch deren Relevanz im Rahmen digitaler Spurensicherung nachvollzogen werden können. Ein besonderer Fokus liegt auf der Erhebung dieser Daten mittels verschiedener Extraktionsmethoden. Für die jeweils betrachteten Datenquellen des vorliegenden Untersuchungsobjektes sollen geeignete methodische Ansätze zur Sicherung und anschließenden Analyse der gewonnenen Informationen aufgezeigt werden. Die Analyse soll letztlich sicherheitsrelevante Erkenntnisse in Bezug auf kritische Ereignisse, etwa im Kontext einer Straftat, ermöglichen und somit einen Beitrag zur forensischen Untersuchung leisten.

Daraus lassen sich die folgenden Forschungsfragen ableiten:

1) Wie lassen sich aus den während der Nutzung eines Yale Linus® Smart Lock erzeugten Netzwerk-, Cloud- und App-Daten digitale Spuren identifizieren, die forensisch relevant sind, und welchen Beitrag leisten diese zur Untersuchung sicherheitsrelevanter Vorfälle?
2) Welche Informationen lassen sich aus der zugehörigen Yale Home App extrahieren, über die die Aktionen gesteuert werden?

## 1.4 Abgrenzung

Die vorliegende Arbeit fokussiert sich auf eine qualitative, fallbasierte Analyse, da die forensische Untersuchung im Mittelpunkt steht und sich durch diese Methodik am zielführendsten umsetzen lässt. Der Untersuchungsgegenstand ist die forensische Verwertbarkeit der erhobenen Daten. Aspekte der Verschlüsselung werden dabei berücksichtigt. Angriffsszenarien hingegen wurden bewusst ausgeklammert, da diese nicht dem Forschungsrahmen dieser Arbeit entsprechen.

Die Analyse beschränkt sich auf das Yale Linus® Smart Lock als konkretes Fallbeispiel. Andere smarte Türschlösser oder IoT-Geräte werden nicht einbezogen, um eine detaillierte Untersuchung des gewählten Geräts zu gewährleisten. Die Datenerhebung erfolgte an einem einzigen Tag. Eine längerfristige Beobachtung über mehrere Tage oder Wochen konnte aufgrund des vorgegebenen Zeitrahmens nicht realisiert werden.

## 1.5 Aufbau der Arbeit

In Kapitel 2 werden die theoretischen und technischen Grundlagen behandelt, die für das Verständnis der Arbeit erforderlich sind. Das Kapitel ist in mehrere Abschnitte gegliedert: Zunächst werden zentrale Begriffe definiert und der Forschungsstand dargestellt, um bisherige Entwicklungen und bestehende Ansätze im Untersuchungsfeld einzuordnen. Anschließend

werden Aufbau, Einrichtung, Funktionen und Nutzerverwaltung des Yale Linus® Smart Schlosses beschrieben. Abschließend erfolgt die Darstellung relevanter technischer Grundlagen, einschließlich der Architekturkonzepte und der verwendeten Technologien, welche die Funktionsweise des betrachteten Systems erläutern.

In Kapitel 3 wird die Methodik der Untersuchung beschrieben. Es werden das Forschungsdesign und die gewählte Herangehensweise erläutert, die eingesetzten Erhebungsmethoden und deren Durchführung veranschaulicht sowie das methodische Vorgehen und die verwendeten Werkzeuge zur Datenerhebung und -analyse beschrieben.

In Kapitel 4 werden die Ergebnisse systematisch dargestellt. Es wird aufgezeigt, welche Daten im Rahmen der Untersuchung erhoben wurden und wie sie in Bezug auf die Forschungsfragen strukturiert und ausgewertet wurden.

In Kapitel 5 erfolgt die Diskussion der Ergebnisse. Die Resultate werden interpretiert, die forensische Relevanz der Erkenntnisse bewertet, Limitationen und Herausforderungen der Untersuchung aufgezeigt und abschließend ein Ausblick auf zukünftige Forschungs- und Entwicklungsperspektiven gegeben. Das Kapitel schließt mit einem Fazit, in dem die zentralen Erkenntnisse zusammengefasst werden.

# 2 Grundlagen

In diesem Kapitel werden zunächst zentrale Begriffe des Themenfelds IoT erläutert und der aktuelle Forschungsstand im Bereich smarter Türschlösser dargestellt. Anschließend wird die Funktionsweise des Yale Linus® Smart Lock im Detail beschrieben. Dazu gehören sein Aufbau, der sowohl die physische Einheit als auch die mechanischen und elektronischen Komponenten umfasst, die Einrichtung mit den Aspekten der physischen Installation, der digitalen Einrichtung und des Datenschutzes sowie die Funktionen in Bezug auf Aktionen, die getätigt werden können, Nutzerrollen und Zugangsrechte. Darüber hinaus werden die zugrunde liegenden Architekturkonzepte der IoT- und Systemarchitektur sowie die dabei verwendeten Technologien wie Bluetooth Low Energy (BLE), Wireless Fidelity (Wi-Fi), Transport Layer Security (TLS) und Advanced Encryption Standard (AES) erläutert.

## 2.1 Begriffseinordnung

Die Begrifflichkeit Industrie 4.0 bezeichnet die Verknüpfung industrieller Produktionsprozesse mit modernen Techniken der Informationstechnologie, um Automatisierungsvorgänge umfassend zu ermöglichen. Das zentrale Element ist die sogenannte intelligente Vernetzung, die unterschiedliche Dimensionen umfasst: Eine anpassungsfähige und effiziente Produktionsweise, die Fähigkeit von Fabriken, sich flexibel auf neue Anforderungen einzustellen, die Entwicklung individueller und kundenorientierter Lösungen, die Optimierung logistischer Abläufe, die Standardisierung der Datenintegration zwischen Systemen, ein ressourcenschonender und nachhaltiger Umgang mit Materialien sowie die fortschreitende Digitalisierung von Produktionsprozessen. Im Mittelpunkt steht die Verbesserung und Optimierung der Wertschöpfungskette über den gesamten Produktlebenszyklus hinweg. Eine wesentliche Voraussetzung hierfür bildet die durchgängige Daten- und Kommunikationsverbindung von Sensoren auf der Produktionsebene bis hin zu internetbasierten Plattformen. [1]

Der Begriff Internet of Things steht in enger Verbindung mit Industrie 4.0, da beide Konzepte darauf abzielen, Menschen, Maschinen und Produkte miteinander zu verknüpfen. IoT bezeichnet die Einbindung physischer Geräte in ein übergeordnetes, internetbasiertes Gesamtsystem. Zu diesen Geräten zählen Beleuchtungssysteme, Thermostate und allgemein bekannte smarte Haushaltsgeräte, die sich von überall aus steuern lassen. IoT-Geräte erfassen Zustände über Sensoren, werten die Daten aus und führen daraufhin Aktionen über Aktoren aus. [8]

Das Industrial Internet of Things (IIoT) baut auf den grundlegenden Prinzipien des Internet of Things auf, indem physische Geräte vernetzt, Zustände erfasst und über Aktoren Aktionen ausgelöst werden [8]. Im industriellen Kontext, insbesondere im Rahmen von Industrie 4.0 [1], wird IIoT eingesetzt, um Produktionsfunktionen, Produktionsstätten sowie vernetzte Fertigungssysteme zu überwachen und zu steuern [1]. Darüber hinaus ermöglicht IIoT die Optimierung von Produktionsprozessen durch Echtzeit-Datenanalyse und die nahtlose Integration von Maschinen, Anlagen und Logistiksystemen [1].

Smart Home bezeichnet ein Wohnkonzept, das technologische Entwicklungen nutzt, um das subjektive Wohlbefinden, den Schutz des privaten Wohnbereichs sowie das allgemeine Wohnkomfortgefühl zu gewährleisten. Ein weiterer Schwerpunkt liegt auf der Optimierung von Automatisierungs- und Steuerungsoptionen, die durch den Einsatz von IoT-Geräten die Funktionalität und Effizienz dieses Wohnkonzepts sicherstellen. Smart Home umfasst somit die Vernetzung unterschiedlichster Funktionen verschiedener Geräte innerhalb der Wohnumgebung. [9]

Die IoT-Forensik lässt sich als spezialisierter Bereich der Digitalen Forensik verstehen. Ziel beider Disziplinen ist die Identifizierung und gesicherte Extraktion digitaler Informationen, die als gerichtsverwertbare Beweismittel dienen können [10]. Im Vergleich zur klassischen Digitalen Forensik ist der Umfang der potenziellen Informationsquellen in der IoT-Forensik deutlich größer [11], da neben den Endgeräten auch Netzwerk- und Cloud-Systeme relevante Daten generieren, übertragen oder speichern können [10].

In der nachfolgenden Abbildung 2.1 sind diese drei Bereiche grafisch dargestellt. Die Grafik veranschaulicht die Beziehungen zwischen den Komponenten sowie deren jeweilige Schwerpunkte.

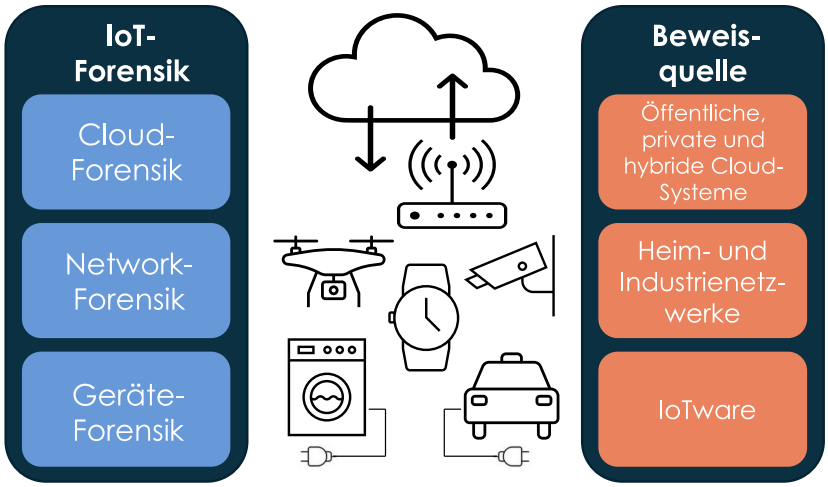



**Abbildung 2.1:** Komponenten der IoT-Forensik, basierend auf: [11]

Ein zentraler Bestandteil der IoT-Forensik ist die Geräte-Forensik, die sich auf die physische Untersuchung von Hardwarekomponenten konzentriert. Hierbei werden bspw. Festplatten untersucht, um detaillierte Informationen über den Nutzer des Geräts zu gewinnen. Ein weiterer wesentlicher Bereich ist die Netzwerkforensik. Jedes IoT-Gerät ist Teil eines Netzwerks, über das Daten mithilfe verschiedener Protokolle übertragen werden. Die Analyse dieser netzwerkbezogenen Informationen ermöglicht es, die Interaktionen und Aktivitäten der angeschlossenen Geräte nachvollziehbar zu rekonstruieren. Darüber hinaus spielt die Cloud-Forensik eine bedeutende Rolle. Viele IoT-Geräte verfügen nur über begrenzte Speicherressourcen, sodass ein Großteil der erzeugten Daten in Cloud-Diensten abgelegt wird. Die Untersuchung dieser Daten erlaubt es, sowohl gerätespezifische Informationen als auch Angaben über die Besitzer zu extrahieren. [7]

Die drei genannten Komponenten der IoT-Forensik (Geräte-, Netzwerk- und Cloud-Forensik) bilden zusammen das Fundament dieser Disziplin. Ihre Betrachtung verdeutlicht, wie breit gefächert die Quellen digitaler Beweise in einer IoT-Umgebung sind. Gleichzeitig wird dadurch die Notwendigkeit spezieller Methoden und Werkzeuge deutlich, um diese Daten zuverlässig zu sichern und auszuwerten. [11]

## 2.2 Forschungsstand

Die forensische Analyse smarter Türschlösser ist ein junges, dennoch immer wichtiger werdendes Forschungsfeld im Bereich des IoT. Aufgrund ihrer digitalen Funktionsweise eröffnen smarte Schlösser nicht nur potenzielle Angriffspunkte für unbefugte Zugriffe, sondern generieren zugleich zahlreiche digitale Spuren, die bei der forensischen Analyse von Sicherheitsvorfällen oder Straftaten von Relevanz sein können.

Der Fachartikel *Forensic Analysis of the August Smart Device Ecosystem* von Shinelle Hutchinson und Umit Karabiyik aus dem Jahr 2020 stellt somit eine zentrale Studie in diesem Bereich dar. Die Studie beschäftigt sich mit der forensischen Sicherung und Analyse des August Smart Lock Pro ASL-03, AC-R1 [12] der August Incorporated (Inc.). Dabei wird die Kommunikation zwischen dem August Smart Lock Pro und der August Home Steuerungs-App unter der Verwendung einer Bluetooth- bzw. Wi-Fi-Verbindung behandelt. Außerdem spielt die Kommunikation der Geräte untereinander eine wichtige Rolle. Darin eingeschlossen sind der Netzwerkverkehr und die verwendeten Verschlüsselungsprotokolle. Für die Rekonstruktion forensischer Artefakte ist zum einen die Nutzung und zum anderen der Zustand des Geräts von großer Bedeutung. Deshalb untersucht die Studie neben einem gejailbreakten iPhone Operating System (iOS)-Gerät sowohl ein gerootetes als auch ein nicht gerootetes Android-Gerät. [13]

Jailbreaking bzw. Rooting bezeichnet das gezielte Freischalten erweiterter Zugriffsrechte auf das Dateisystem eines Geräts. Da dabei der Systemzustand verändert wird, sollte dieser Schritt nur bei zwingender Notwendigkeit erfolgen. [14]

In der Literaturübersicht gibt der Fachartikel eine Übersicht über Fachliteratur, die sich mit dem Bereich der IoT-Forensik beschäftigt und für die vorliegende Untersuchung relevant ist. So betrachteten und analysierten Goudbeek et al. mögliche Beweisdaten und deren Beschaffungsweg im Bereich des Smart Homes. Es ist ihnen gelungen, eine Reihe an Artefakten zu identifizieren, die nach dem Löschen wiederhergestellt werden können. [15]

In *A smart home is no castle: Privacy vulnerabilities of encrypted IoT traffic* untersuchten Apthorpe et al. den Netzwerkverkehr von IoT-Geräten und stellten fest, dass trotz der vollständigen Verschlüsselung des Datenverkehrs signifikante Verkehrsspitzen im Netzwerk beobachtet werden konnten, deren Zeitpunkte mit der Nutzung der Geräte durch die Anwender übereinstimmten [16]. Diese Erkenntnisse nutzten die Autoren des Fachartikels für die Durchführung ihrer Studie an dem August Smart Lock Pro.

**Methodik** Die Studie orientiert sich am forensischen Untersuchungsmodell für die Datenextraktion, welches vom National Institute of Standards and Technology (NIST) empfohlen wird [17]. Dabei kamen drei Smartphones zum Einsatz: Ein iPhone 7 mit iOS 12 sowie zwei Samsung Galaxy S7, die mit den Android-Versionen 7.0 (Nougat) und 8.0 (Oreo) betrieben werden. Die Auswahl dieser Geräte erfolgte, da sie die dominierenden Betriebssysteme im mobilen Bereich abdecken und somit für die Zielgruppe der Studie relevant sind. Zudem zeigte sich, dass diese Modelle ohne größere Schwierigkeiten für die Analyse von Smart Locks gejailbreakt bzw. gerootet werden können. [13]

Für die forensische Datenerfassung kamen verschiedene Softwarelösungen zum Einsatz, darunter Magnet ACQUIRE (Version 2.21.0.18374), Magnet AXIOM (Version 3.7.0.1679) sowie MSAB XRY (Version 7.12.0, Build 25665). AXIOM und XRY sind dabei weit verbreitete Tools, die häufig von Strafverfolgungsbehörden genutzt werden. Die Extraktion der Daten aus den Magnet ACQUIRE-Abbildern erfolgte mit BulkExtractor (Version 1.5.2). Der durch die IoT-Geräte erzeugte Netzwerkverkehr wurde mit Wireshark (Version 3.0) und einem Alfa AWUS1900 Netzwerkadapter erfasst. Die Telefondaten-Analyse erfolgte auf einem Dell Optiplex 7060 mit Windows 10 Education (Version 1809). [13]

**Laboreinrichtung** Für die Durchführung der Studie wurde eine Mini-Smart-Home-Umgebung geschaffen. Diese umfasste einen Alfa AWUS1900 Netzwerkadapter — zur vollständigen Erfassung des Netzwerkverkehrs –, einen TP-Link Archer AC1900 Router sowie die zuvor erwähnten drei Smartphones. Als IoT-Gerät kam das August Smart Lock Pro (Modellnummer ASL-03, AC-R1) zum Einsatz. [13]

**Datenbestand** Im Rahmen der Vorbereitungen wurden alle Smartphones in ihren Auslieferungszustand zurückversetzt sowie neue iCloud- und Google-Konten angelegt. Auf dem iPhone wurde ein neues iCloud-Konto eingerichtet und aktiviert, während die Galaxy S7-Geräte mit den neu erstellten Google-Konten verknüpft wurden. Danach wurde von jedem Gerät mithilfe von Magnet AXIOM Process eine 1:1-Kopie erstellt und die Mobilgeräte nach denen des NIST vorgeschlagenen Richtlinien zur Durchführung der Studie vorbereitet. Dazu gehörten die Installation der August Home App sowie die Erstellung je eines August-Kontos auf dem iPhone und dem Galaxy S7 mit der Version 7.0 (Nougat). Beide Geräte wurden anschließend zur Interaktion mit dem August Smart Lock Pro verwendet. Hierfür wurde das Schloss zunächst gemäß den Anweisungen innerhalb der App eingerichtet und anschließend getestet. Nach erfolgreicher Kopplung mit dem Smartphone erfolgte die Steuerung des Schlosses über die App. Im Einzelnen wurden folgende Funktionen getestet: Ent- und Verriegeln des Schlosses, die Aktivierung und der Test der automatischen Sperrfunktion (nach 30 Sekunden), sowie die Verwendung der Auto-Unlock-Funktion. Zusätzlich wurde einem weiteren Gerät, dem Galaxy S7 mit der Android-Version 8.0 (Oreo) ein virtueller Schlüssel zugewiesen. Mit diesem Gerät wurde das Smart Lock ebenfalls bedient. Im Anschluss wurde der virtuelle Schlüssel wieder entzogen und es wurde geprüft, ob weiterhin eine Interaktion mit dem Schloss möglich war. Während der Nutzung wurde der Netzwerkverkehr durchgehend aufgezeichnet. Nach Abschluss dieser Tests folgte eine zweite Erfassung der Geräte. Danach wurden das Galaxy S7 (Version 7.0) gerootet und das iPhone gejailbreakt, um weiterführende Datenanalysen zu ermöglichen. Abschließend wurde eine vollständige forensische Sicherung durchgeführt. Die Auswertung erfolgte mit Magnet AXIOM Examiner und MSAB XRY. [13]

**Ergebnisse** In Tabelle 2.1 sind zum Vergleich die wiederherstellbaren Artefakte eines nicht gerooteten sowie eines gerooteten Android-Geräts übersichtlich dargestellt. Zu den erfassten Benutzerdaten des gerooteten Geräts zählen persönliche Informationen wie Name, E-Mail-Adresse und Telefonnummer, das Profil- sowie das Hausbild des Benutzers, ergänzt durch die jeweilige August-Identifikationsnummer (ID), die einem spezifischen Nutzerprofil zugeordnet ist. August bietet die Auto-Unlock-Funktion an, welche auf Geofence-Technologie basiert. Mithilfe dieser wird festgelegt, in welchem Umkreis das Smart Lock automatisch entriegelt wird. Bei Aktivierung der Funktion werden die aktuellen Standortdaten des Nutzers in Form von Global Positioning System (GPS)-Koordinaten in einer entsprechenden Datenbank gespeichert. Diese GPS-Daten dokumentieren ausschließlich die räumlichen Positionsveränderungen des Nutzers. Darüber hinaus dokumentieren die Geräteinformationen sämtliche Interaktionen mit den August-Geräten. Hierzu zählen unter anderem Aktionen wie das (Ent)sperren, automatische Entsperrvorgänge, Klingelereignisse sowie Bewegungserkennungen. Die Speicherorte der eben genannten wiederherstellbaren Artefakte auf dem gerooteten Android-Gerät sind in Tabelle 2.2 je nach Image aufgelistet. Der Datenverkehr von August im Kontext des Android-Geräts erfolgt verschlüsselt, entweder über das Transmission Control Protocol (TCP) oder mittels TLS in der Version 1.2. [13]

**Tabelle 2.1:** Wiederherstellbare Artefakte auf gerooteten bzw. nicht gerooteten Android-Geräten [13]

| Artefakt | Nicht gerootet | Gerootet |
|---|---|---|
| Benutzerdaten | Nein | Ja |
| Benutzerpasswort | Nein | Ja |
| Benutzer-GPS-Daten | Nein | Ja |
| Geräteinteraktionen | Nein | Ja |

**Tabelle 2.2:** Speicherorte der Artefakte auf dem gerooteten Android-Gerät [13]

| Artefakt | MSAB XRY Image | Magnet ACQUIRE Image |
|---|---|---|
| Benutzerdaten | /data/data/com.august.luna/databases/MediaFileDatabase.db<br>/data/data/com.august.luna/files/logs | /data/fab/db.txt<br>/data/logs/db.txt |
| Benutzer-GPS-Daten | /data/data/com.august.luna/databases/LocationDatabase.db | N/A |
| Geräteinteraktionen | /data/data/com.august.luna/databases/MediaFileDatabase.db<br>/data/data/com.august.luna/files/logs | /data/logs/db.txt<br>/data/logs/log.txt |

Das vorgestellte Paper liefert damit einen wertvollen Bezugsrahmen für die vorliegende Arbeit. Durch die forensische Analyse eines vergleichbaren Smart Locks auf Netzwerk-, Cloud- und Appebene schafft es eine methodische Grundlage und verdeutlicht zugleich, an welchen Stellen gezielt nach digitalen Spuren gesucht werden kann. Diese Erkenntnisse bilden eine wichtige Grundlage für die Untersuchung des Yale Linus® Smart Lock und helfen, die eigene Analyse systematisch und vergleichbar einzuordnen.

## 2.3 Aufbau des Yale Linus® Smart Lock

Das Yale Linus® Smart Lock [18] vereint moderne Sicherheitstechnologien mit klassischer Schließmechanik in einem kompakten und funktional durchdachten Design. Sein Aufbau verdeutlicht, dass eine intelligente Steuerung und eine benutzerfreundliche Handhabung gewährleistet werden können, ohne dabei die physische Sicherheit zu vernachlässigen. Um sicherzustellen, dass das System flexible Zugangskontrollen bietet und vor äußeren Gewalteinwirkungen schützt, müssen mechanische und elektronische Komponenten aufeinander abgestimmt werden. Der technische Aufbau bildet somit die Grundlage für die zahlreichen Eigenschaften und Sicherheitsmerkmale, die das Schloss charakterisieren und von herkömmlichen Schließsystemen unterscheiden. In den nachfolgenden Unterkapiteln wird daher jeweils ein Überblick über die physische Einheit sowie die mechanischen und elektronischen Komponenten des Schlosses gegeben.

### 2.3.1 Physische Einheit

Die physische Einheit des Yale Linus® Smart Schlosses besteht aus einem robusten, aluminiumverstärkten Kunststoffgehäuse. Mit Abmessungen von $150\,\mathrm{mm}$ x $58\,\mathrm{mm}$ x $60\,\mathrm{mm}$ in Länge, Tiefe und Höhe sowie einem Gewicht von 770 g weist es insgesamt eine kompakte Bauweise auf. Die vom Hersteller vergebene Gerätenummer lautet *8750001828*. Während *5052847110546* die European Article Number (EAN) (dt.: Europäische Artikelnummer) darstellt. Das Schloss wird zusammen mit vier AA-Alkaline-Batterien (siehe Unterabschnitt 2.3.2), einer Montageplatte, einem Quickstartquide (dt.: Schnellstartanleitung) sowie einem Innensechskantschlüssel geliefert. In dem Lieferumfang ist außerdem die Yale Connect Wi-Fi-Bridge inbegriffen. Diese hat eine Versorgungsspannung von $230\,\mathrm{V}$ ~/ $50\,\mathrm{Hz}$. Die wesentlichen Informationen sind zur besseren Übersicht in Tabelle 2.3 zusammengefasst. [18]

**Tabelle 2.3:** Allgemeine Informationen zum Yale Linus® Smart Schloss [18]

| Angabe | Beschreibung |
|---|---|
| Gerätebezeichnung | Yale Linus® Smart Lock |
| Gerätenummer | 8750001828 |
| EAN-Nummer | 5052847110546 |
| Abmessungen | 150 mm × 58 mm × 60 mm |
| Gewicht | 770 g |
| Lieferumfang | AA-Alkaline-Batterien,<br>Montageplatte,<br>Quickstartguide,<br>Yale Connect WiFi-Bridge |
| Versorgungsspannung Wi-Fi-Bridge | 230 V ∼ / 50 Hz |

Das Gerät wird auf der Innenseite der Tür montiert und ist somit von außen nicht sichtbar. Dies bietet zum einen Schutz gegen Vandalismus und zum anderen sind Angriffspunkte wie das Aufhebeln, Bohren oder anderweitige materielle Manipulationen ausgeschlossen.

### 2.3.2 Mechanische Komponenten

Das Schloss verfügt über einen motorisierten Schließmechanismus, der das Ver- und Entriegeln dessen automatisch steuert [19]. Der Elektromotor ist in dem Schlossgehäuse integriert und arbeitet mit dem vorhandenen Zylinder zusammen, sodass der originale Schlüssel weiterhin verwendet werden kann [19]. Die Energieversorgung erfolgt über Batterien [19]. Darauf wird im folgenden Kapitel noch einmal genauer eingegangen. Durch das automatische Öffnen und Schließen des Schlosses über den integrierten Motor wird der Prozess des manuellen Drehens des klassischen Schlüssels durch den Benutzer ersetzt und der Vorgang gestaltet sich als sehr komfortabel. Daraus kann abgeleitet werden, dass der Benutzer den Schlüssel nicht mehr manuell drehen muss und der Vorgang insgesamt deutlich komfortabler gestaltet ist.

Die Steuerung des Schlosses erfolgt im Regelfall über die zugehörige Smartphone-App [19]. Für den Fall eines technischen Defekts wird empfohlen sicherheitsrelevante Vorkehrungen zu treffen, die bereits vor der Installation berücksichtigt werden müssen. Von zentraler Bedeutung ist hierbei die Verwendung eines Europrofilzylinders mit Not- und Gefahrenfunktion, damit die Kompatibilität von Schloss und Zylinder gegeben ist [20]. Die Not- und Gefahrenfunktion ist ein Sicherheitsmechanismus, der gewährleistet, dass der Zylinder auch dann bedienbar bleibt, wenn sich auf der Gegenseite bereits ein Schlüssel im Schloss befindet [20]. Daraus kann abgeleitet werden, dass eine manuelle Notbedienung jederzeit möglich ist, ohne dass mechanische Änderungen am bestehenden Schloss erforderlich sind. Auf detaillierte Informationen der Montagemöglichkeiten des Geräts wird in Unterabschnitt 2.4.1 genauer eingegangen.

### 2.3.3 Elektronische Komponenten

Um die Funktionsfähigkeit des Schlosses sicherzustellen, ist eine zuverlässige Stromversorgung von großer Bedeutung. Beim Yale Linus® Smart Lock wird die Stromversorgung sichergestellt, indem zwei AA-Alkaline-Batterien in das Gerät eingelegt werden. Die Batterielaufzeit hängt vom Gebrauch des Geräts ab und liegt daher in einer Spanne von sechs bis neun Monaten. Die nachstehenden Faktoren sind dafür ausschlaggebend:

- Anzahl der Schließvorgänge
- Anzahl der Öffnungen pro Tag
- Leichtigkeit der Türöffnung

Wenn die Batterieleistung gering ist, sendet das System eine Nachricht, um den Nutzer darauf aufmerksam zu machen, dass ein baldiger Batteriewechsel erforderlich ist. [21]

Zudem greift die Technologie auf ein System mit Sensoren zurück. Die Funktionsweise der DoorSense™-Technologie (dt.: Türsensor) [21] wird deshalb im Kapitel 2.5 genauer behandelt.

In einem Teardown-Bericht (dt.: Zerlegung) von Tarlogic Security wird die Verwendung des Mikrocontrollers vom Typ DA14680A im Yale Linus® Smart Lock beschrieben [22]. Der Mikrocontroller basiert auf der ARM Cortex-M0-Architektur und verwendet einen 32-Bit-Prozessor [23]. Das führt zu einer sehr energieeffizienten Arbeitsweise des Geräts. Dies ist ideal für batteriebetriebene IoT-Geräte, wie das der Arbeit zugrundeliegende Schloss von Yale [23]. In dem Mikrocontroller ist ein BLE 4.2 Modul mit $2.4\,\mathrm{GHz}$ integriert [22]. So kann die direkte Kommunikation mit dem Smartphone über die Yale Home App ermöglicht werden. Weitere zentrale Funktionen des Schlosses, die über den DA14680A gesteuert werden, sind die Motorsteuerung für das Ver- und Entriegeln sowie die Anbindung an verschiedene Sensoren [22]. Der Mikrocontroller ist in einem Ball Grid Array-Gehäuse untergebracht, was das Auslesen ohne spezielle Techniken wie „Chip-Off“ erschwert. [22] Dies ist eine Methode, bei der Speicherchips aus physischen Geräten entfernt werden, um Daten zu extrahieren.

Im Falle eines Stromausfalls ist es möglich, das Gerät mithilfe eines physischen Schlüssels manuell zu öffnen, da im Vorfeld der Installation sicherheitsrelevante Vorkehrungen getroffen werden müssen, wie bereits in Kapitel 2.3.2 beschrieben.

## 2.4 Einrichtung des Yale Linus® Smart Lock

Für die Einrichtung und anschließende Verwendung des Yale Linus® Smart Schlosses sind zwei Komponenten von wesentlicher Bedeutung. Zum einen die physische Installation des Gerätes am standardmäßig vorhandenen Türschloss und zum anderen die digitale Einrichtung der App auf dem Smartphone.
Ersteres beschäftigt sich damit, herauszufinden, ob der Zylinder des vorliegenden Schlosses mit dem Yale Linus® Smart Schloss kompatibel ist. Dafür liegt ein Kompatibilitätscheck zugrunde. Aufgrund dieses Resultates ergeben sich Montagemöglichkeiten für:

- kompatible Zylinder
- bündige Zylinder
- einstellbare Zylinder

Letzteres gibt Auskunft darüber, welche Hardware und Software für die Nutzung des smarten (dt.: intelligenten) Schlosses verwendet werden und wie die Vorgehensweise bei der Einrichtung der Yale Home App ist. Außerdem wird genauer darauf eingegangen, welche Daten bei der Einrichtung der App automatisch erfasst sowie vom Benutzer eigenständig übermittelt werden. In den nachfolgenden Unterkapiteln wird daher ein Überblick darüber gegeben.

### 2.4.1 Physische Installation

Die Montage des Yale Linus® Smart Schlosses erfolgt auf der Innenseite der Wohnungstür. Damit wird sichergestellt, dass keine gewaltsamen Einwirkungen von außen stattfinden und die physische Komponente des Schlosses unbeschädigt bleibt.
Jedoch sind vor der Anbringung des smarten Schlosses weitere Maßnahmen zu treffen, um den Ansprüchen der Verwendung dessen gerecht zu werden. Wie in den vorherigen Unterkapiteln 2.3.2 und 2.3.3 bereits erwähnt, müssen sicherheitsrelevante Vorkehrungen getroffen

werden und das vorhandene Zylinderschloss bestimmten Anforderungen entsprechen. Dadurch soll die Funktionalität sowohl bei einem technischen Defekt als auch bei einem Stromausfall sichergestellt werden. Daher muss das Zylinderschloss auf Kompatibilität geprüft werden.

Die ASSA ABLOY Sicherheitstechnik GmbH, zu deren Markenportfolio auch Yale gehört [24], veröffentlichte dazu im Jahr 2020 ein Video, das potenzielle Kunden darüber aufklärt, welche technischen Voraussetzungen das vorhandene Türschloss für die Nutzung des Yale Smart Locks erfüllen muss. Daraus ist abzuleiten, dass das bestehende Zylinderschloss den Anforderungen entspricht, wenn:

- auf der Innen- und Außenseite gleichzeitig ein Schlüssel verwendet werden kann bzw. ein Europrofilzylinder mit Not- und Gefahrenfunktion verbaut ist
- auf der Türinnenseite ein Türgriff und kein Drehknauf am Zylinder befestigt ist
- der Schlüssel maximal $6\,\mathrm{mm}$ dick ist
- der Schlüssel maximal $40\,\mathrm{mm}$ aus dem Zylinder übersteht [25]

In Abbildung A.1 ist ein binärer Entscheidungsbaum beigelegt, welcher auf Grundlage des Kompatibilitätschecks der Yale Home-Webseite erstellt wurde. Dieser hilft zu bestimmen, ob das vorhandene Zylinderschloss kompatibel ist oder ein Ersatz- bzw. Einstellzylinder erforderlich wird. Zudem erweitert er die oben stehende Liste der Anforderungen um den folgenden Punkt: Der Abstand zwischen der Zylindermitte und dem Türrahmen muss mindestens $35\,\mathrm{mm}$ betragen [24].

Nachdem das vorhandene Zylinderschloss geprüft und alle Anforderungen erfüllt wurden, kann das Yale Linus® Smart Lock montiert werden. Für Zylinder, die $3\,\mathrm{mm}$ oder mehr über die Türgarnitur herausstehen, ist der Schritt-für-Schritt-Anleitung für die Installation eines kompatiblen Zylinders zu folgen. Die Installation kompatibler Zylinder, die weniger als $3\,\mathrm{mm}$ aus der Türgarnitur herausragen, ist in der Anleitung für bündige Zylinder beschrieben. Ergibt der Kompatibilitätscheck, dass der bereits vorhandene Zylinder nicht kompatibel ist, muss dieser durch einen anderen Zylinder ersetzt werden. Entscheidet sich der Käufer in diesem Fall für einen einstellbaren Zylinder von Linus, ist der Montageanleitung zur Installation eines einstellbaren Zylinders zu folgen.

### Kompatibler Zylinder

| Installation eines kompatiblen Zylinders [26] | |
|---|---|
| **Schritt 1:** | Die Linus-Einheit aus der Verpackung entnehmen. Anschließend die Flügel öffnen und die Montageplatte entfernen. |
| **Schritt 2:** | Hinzufügen der Montageplatte auf gegebener Zylinderinnenseite. Möglicherweise ist es notwendig, die Schrauben der Montageplatte um den Zylinder herum zu lösen, um die Platte richtig anbringen zu können. Festziehen der Schrauben, wenn die Platte den Zylinder berührt. Dafür ist der Umverpackung ein Inbusschlüssel beigefügt. |
| **Schritt 3:** | Den Schlüssel in den Zylinder schieben (auf der Innenseite). |

**Schritt 4:** Die Linus-Einheit auf der Montageplatte befestigen. Dafür als Erstes die Einheiten-Oberseite auf die Montageplatte schieben.
**Schritt 5:** Die Linus-Einheit positionieren, indem die Flügel dieser geschlossen werden.
**Schritt 6:** Ablösen der Schutzfolie von dem DoorSense™-Magnet. Den Magnet an der Türzarge, in Höhe des Logos von Yale, anbringen.

### Bündiger Zylinder

Installation eines bündigen Zylinders [26]

**Schritt 1:** Die Linus-Einheit aus der Verpackung entnehmen. Anschließend die Flügel öffnen und die Montageplatte entfernen.
**Schritt 2:** Die Schutzfolie von dem Klebeband abziehen.
**Schritt 3:** Das Klebeband an der Montageplattenrückseite fixieren.
**Schritt 4:** Die von außen an das Klebeband angebrachte Schutzfolie mithilfe der angebrachten Lasche entfernen.
**Schritt 5:** Ausrichten und Anbringen der Montageplatte auf der Zylinderinnenseite. Möglicherweise ist es notwendig, die Schrauben der Montageplatte um den Zylinder herum zu lösen, um die Platte richtig anbringen zu können. Festziehen der Schrauben, wenn die Platte den Zylinder berührt. Dafür ist der Umverpackung ein Inbusschlüssel beigefügt.
**Schritt 6:** Die Linus-Einheit auf der Montageplatte anbringen. Dafür als Erstes die Einheiten-Oberseite auf die Montageplatte schieben.
**Schritt 7:** Die Linus-Einheit positionieren, indem die Flügel dieser geschlossen werden.
**Schritt 8:** Ablösen der Schutzfolie von dem DoorSense™-Magnet. Den Magnet an der Türzarge, in Höhe des Logos von Yale, anbringen.

### Einstellbarer Zylinder

Installation eines einstellbaren Zylinders [27]

**Schritt 1:** Vor Beginn auf der Innenseite die Länge des herausstehenden Zylinders messen. Ergibt die Messung weniger als $3\,\mathrm{mm}$, ist das für den Schritt 3a von Bedeutung.
**Schritt 2:** Entfernen des vorhandenen Zylinders. Die befestigende Schraube am bestehenden Europrofilzylinder unter dem Schließriegel lösen. Den Schlüssel in den Zylinder schieben und um ca. 15 Grad drehen, um diesen aus der Tür zu entfernen.
**Schritt 3:** Die Länge des bestehenden Schließzylinders bestimmen. Hierzu wird die Länge von beiden Hälften bis zur Zylindermitte (Öffnung für die Befestigungsschraube) bestimmt. Das Innen- und Außenmaß notieren, um später den einstellbaren Zylinder passend einzustellen.
**a)** Wenn der in Schritt eins vorhandene Zylinder auf der Türinnenseite keine $3\,\mathrm{mm}$ hervorgeragt hat, müssen zu dem Innenmaß $3\,\mathrm{mm}$ addiert werden. Damit kann das Yale Linus® Smart Lock unkompliziert befestigt werden.

**b)** Ergibt die Messung sowohl eine Außen- als auch eine Innenlänge von $30\,\mathrm{mm}$, muss nichts verändert werden, da der einstellbare Zylinder passgenau ist und montiert werden kann. Es kann bei Schritt elf weitergemacht werden.

**Schritt 4:** Den einstellbaren Zylinder von Linus entsprechend der Länge des vorhandenen Zylinders justieren. Herstellen der genauen Längen mithilfe der externen und internen Abstandhalter.

**Schritt 5:** Auswählen des erforderten externen Abstandhalters und der externen Frontplatte. Liegt die Außenlänge des vorhandenen Zylinders zwischen $35\,\mathrm{mm}$ und $45\,\mathrm{mm}$, muss die kürzere Frontplatte verwendet werden. Beträgt die Länge $50\,\mathrm{mm}$ bis $60\,\mathrm{mm}$, sollte die längere Frontplatte gewählt werden.

**Schritt 6:** Die unterhalb des Zylinders befestigten Schrauben mit dem Inbusschlüssel lösen und die zwei Füllstücke entnehmen.

**Schritt 7:** Die externen Abstandhalter zusammen mit der Frontplatte auf dem einstellbaren Zylinder anbringen. Dem Zylinder sollte der kleinste externe Abstandhalter am nächsten sein. Die Abstandhalter unten am Zylinder festschrauben.

**Schritt 8:** Sobald die Frontplatte und die externen Abstandhalter montiert sind, wird der Schlüssel von außen in den einstellbaren Zylinder gesteckt. Danach prüfen, ob sich der einstellbare Zylinder frei drehen lässt.

**Schritt 9:** Beträgt das Innenmaß des vorhandenen Zylinders, inklusive der $3\,\mathrm{mm}$ aus dem dritten Schritt, mehr als $30\,\mathrm{mm}$, werden die internen Abstandhalter genutzt, um die Differenz zwischen $30\,\mathrm{mm}$ und der Innenlänge des Zylinders zu bilden.

**Schritt 10:** Zusammensetzen der Innenseite des einstellbaren Zylinders mit den korrekten Abstandhaltern der Tür. Die Größenangabe ist auf der Seite der Abstandhalter zu finden. Anbringen der Abstandhalter im mittleren Loch des Zylinders. Der einstellbare Zylinder und der kleinere Abstandhalter sollten in unmittelbare Nähe zueinander positioniert werden.

**Schritt 11:** Nachdem der verstellbare Zylinder montiert wurde, wird die Heckstange hineingeschoben. Bei einem Innenmaß von maximal $45\,\mathrm{mm}$, ist die kürzere Heckstange zu benutzen. Bei einem größeren Innenmaß wird die längere Heckstange verwendet. Anschließend überprüfen, ob sich der verstellbare Zylinder frei drehen lässt.

**Schritt 12:** Die Heckstange auf der Türinnenseite solange rotieren, bis der Schließbart und das Gehäuse des einstellbaren Zylinders auf einer Linie liegen und die Öffnung im Türblatt übereinstimmt. Möglicherweise ist es notwendig die Heckstange etwas hin und her zu bewegen, damit der Zylinder richtig sitzt. Es kann sein, dass die bestehende Türgarnitur etwas gelockert werden muss, falls diese zu eng um den verstellbaren Zylinder befestigt ist. Den verstellbaren Zylinder mithilfe der Befestigungsschraube positionieren und die Türgarnitur festziehen.

**Schritt 13:** Die Linus-Einheit aus der Verpackung entnehmen. Anschließend die Flügel öffnen und die Montageplatte entfernen.

**Schritt 14:** Hinzufügen der Montageplatte auf gegebener Zylinderinnenseite. Möglicherweise ist es notwendig, die Schrauben der Montageplatte um den Zylinder herum zu lösen, um die Platte richtig anbringen zu können. Festziehen der Schrauben, wenn die Platte den Zylinder berührt. Darüber hinaus ist notwendig, dass die $3\,\mathrm{mm}$ zur Verfügung stehen. Dafür ist der Umverpackung ein Inbusschlüssel beigefügt

**Schritt 15:** Auf der Innenseite des einstellbaren Zylinders die Heckstange anfügen. Überprüfen, ob sich der verstellbare Zylinder frei drehen lässt. Bei einem Innenmaß von maximal $45\,\mathrm{mm}$ die kurze Heckstange benutzen. Bei einem größeren Maß auf die längere Heckstange zurückgreifen.

**Schritt 16:** Die Linus-Einheit auf der Montageplatte befestigen. Dafür als Erstes die Einheiten-Oberseite auf die Montageplatte schieben.

**Schritt 17:** Die Linus-Einheit positionieren, indem die Flügel dieser geschlossen werden.

**Schritt 18:** Ablösen der Schutzfolie von dem DoorSense™-Magnet. Den Magnet an der Türzarge, in Höhe des Logos von Yale, anbringen.

### 2.4.2 Digitale Einrichtung

Für die uneingeschränkte Nutzung des Schlosses und der Inbetriebnahme der App wird entweder ein Smartphone mit Android- oder iOS-Betriebssystem benötigt. Dabei sollte berücksichtigt werden, dass die Anwendung nur für Geräte mit Android 8.0 und integriertem Bluetooth 4.0 verfügbar ist und ausschließlich auf Samsung, Google und High Tech Computer Corporation Geräten getestet wird. Kompatibel sind ebenfalls alle Geräte mit iOS 14 oder höher. Daraufhin ist es möglich, die Yale Home App im offiziellen App-Store der entsprechenden Betriebssystemanbieter herunterzuladen. [28]

Der erste Schritt nach Installation der App ist die Erstellung eines Yale Home-Kontos (Abbildung A.2a). Dafür müssen die Nutzungs- und Produktbedingungen sowie die Datenschutzerklärung akzeptiert werden (Abbildung A.2b). Anschließend müssen Name, Vorname, E-Mail-Adresse, Telefonnummer und Land angegeben sowie ein Passwort gewählt werden (Abbildung A.2c). Das Passwort muss eine Mindestlänge von acht Zeichen aufweisen sowie jeweils einen Klein- und Großbuchstaben, eine Ziffer und ein Sonderzeichen beinhalten. Nach der Angabe der persönlichen Daten und der Passwortwahl wird ein Short Message Service-Verifikationscode an die angegebene Telefonnummer versendet. Dieser muss im nächsten Schritt in das dafür vorgesehene Feld eingegeben und bestätigt werden (Abbildung A.2d). Damit ist die Einrichtung des Kontos abgeschlossen (Abbildung A.2e). Anschließend daran öffnet sich das Hauptmenü, in dem das Hinzufügen neuer Geräte möglich ist (Abbildung A.3a). Nach dem Einlegen der mitgelieferten Batterien wird das Yale Linus® Smart Lock automatisch erkannt (Abbildung A.3b). Im Anschluss daran werden dem Nutzer drei Fragen gestellt, die darauf abzielen, die Art des Türschlosses festzustellen und die Hardware zu ermitteln (Abbildung A.4). Die Fragen beziehen sich auf:

- den Besitz eines Europrofilzylinders (Abbildung A.5a)
- die Möglichkeit, die Tür von außen per Drücker zu betätigen (Abbildung A.5b)
- die Notwendigkeit den Türgriff zum Verriegeln hochzuklappen (Abbildung A.5c) [29]

Nach Abschluss der Befragung kann das Schloss mithilfe einer Installationseinweisung, die innerhalb der Anwendung bereitgestellt wird, eingebaut werden (Abbildung A.6). Hinweise und Schritt-für-Schritt-Anleitungen für den Einbau des Schlosses unter Beachtung aller Grundvoraussetzungen sind in Unterabschnitt 2.4.1 aufgeführt. Daraufhin wird das Gerät eingerichtet, indem diesem sowohl ein Name für das Zuhause als auch für die genutzte Tür vergeben werden (Abbildung A.7). Damit kann bei der Verwendung mehrerer Schlösser der Überblick behalten werden. Danach erfolgt, falls notwendig, die Installation eines Firmware-Updates.(Abbildung A.8). Im Anschluss daran wird die DoorSense™-Technologie, die den Zustand der Tür als offen oder geschlossen erkennen soll, angebracht und anschließend mittels der Installationsanweisungen in Betrieb genommen (Abbildung A.9a). Der DoorSense™-Magnet muss vor der Nutzung kalibriert werden, um den Türzustand fehlerfrei erkennen zu können (Abbildung A.9b). Zum Schluss muss das Schloss mit dem Wireless Local Area Network (WLAN) (dt. drahtloses lokales Netzwerk) verbunden werden. Dafür wird die Yale Connect Wi-Fi-Bridge (dt.: Netzwerkbrücke) verwendet und eingerichtet. Vor Beginn der Einrichtung ist die Verbindung des Telefons mit dem WLAN sicherzustellen. Anschließend muss die Bridge in der Nähe des Schlosses an den Strom angeschlossen werden. Hierfür empfiehlt sich eine Steckdose im Umkreis von ca. drei Metern. Danach soll die Taste auf der Vorderseite der Bridge fünf Sekunden lang gedrückt werden bis der darüberliegende Punkt anfängt langsam grün zu blinken. Das ist das Zeichen dafür, dass das Gerät erfolgreich am Strom angeschlossen ist und mithilfe des WLANs mit dem Heimnetzwerk verbunden werden kann. Im nächsten Schritt wird das zu nutzende WLAN ausgewählt und die Yale Home-App verbindet sich darüber temporär mit der Bridge. Die Ausführung eines Signal-Tests und das automatische Durchführen eines Schließvorgangs runden die Einrichtung der Wi-Fi-Bridge ab. Die Schritte können in den Abbildungen A.10 und A.11 nachvollzogen werden. [29]

### 2.4.3 Datenschutz

In der Datenschutzerklärung, die für alle Yale-Produkte gilt, erläutert das Unternehmen zunächst, welche Nutzerdaten von Eigentümern für die Verwendung der Produkte erfasst werden. Dazu zählen der Name, die angegebene E-Mail-Adresse und Telefonnummer, das vergebene Passwort, eine Produktkennung sowie bestimmte Identifikationsmerkmale zur Überprüfung der Zugangsberechtigung. Darüber hinaus werden für eingeladene Gäste, deren Mobilfunknummern sowie weitere vom Eigentümer bereitgestellte Angaben wie Name oder E-Mail-Adresse gespeichert. [30]

Für den Fall, dass Funktionen wie z. B. Auto Unlock aktiviert sind und der Zugriff auf den Standort gewährt wurde, verbleiben diese Daten ausschließlich auf dem Gerät des Eigentümers und werden nicht an Yale übermittelt. [30]

Automatisch erhobene System- und Nutzungsinformationen umfassen unter anderem: Eine eindeutige Nutzerkennung, Interaktionen mit den bereitgestellten Diensten, den Funktionsstatus der Produkte, Aktivitätsprotokolle einschließlich der Einrichtung von Gastzugängen sowie technische Daten zu den verwendeten Geräten, wie das Betriebssystem, die App-Version, die Seriennummer, die Media Access Control (MAC)-Adresse, die Softwareversion und Installationsparameter. [30]

Bei der Nutzung der Yale Home App werden dieselben Datenarten erfasst, wie sie im ersten Absatz allgemein für die Nutzung der Produkte beschrieben sind. Ergänzend verarbeitet das Unternehmen weitere Informationen, darunter Gerätekennungen, Internet Protocol (IP)-Adressen, Aktivitätsaufzeichnungen sowie Vorschaubilder und Videoaufzeichnungen, sofern diese über die App abgerufen werden. Die letztgenannten Datenarten fallen bei der Verwendung des Yale Linus® Smart Locks allerdings nicht an. [30]

Die Gründe für die Speicherung und Verarbeitung dieser Angaben sind vielfältig. Zum einen dienen sie dazu, Kunden die erwarteten Dienstleistungen bereitzustellen – hierzu gehören die Identifizierung, die Bereitstellung von Software-Updates sowie die Gewährleistung der Funktionsvielfalt innerhalb der Yale App. Ebenso wird durch die Datenspeicherung die Bereitstellung von Supportleistungen ermöglicht. Zum anderen sind diese Informationen erforderlich, um die Sicherheit der Dienste zu wahren, etwa durch den Schutz vor Schadsoftware, Betrugsversuchen oder unbefugtem Zugriff. [30]

Von besonderer Relevanz für forensische Fragestellungen ist schließlich der Umstand, dass auch die Einhaltung gesetzlicher Vorschriften eine Rolle spielt. Yale ist verpflichtet, bestimmte Daten offenzulegen, sodass Ermittlungsbehörden unter Umständen auf relevante, möglicherweise belastende Informationen zugreifen können, wenn dies für die Strafverfolgung erforderlich ist. [30]

## 2.5 Funktionen und Nutzerverwaltung des Yale Linus® Smart Lock

Das Yale Linus® Smart Lock erweitert ein klassisches Türschloss um digitale Steuerungs- und Überwachungsfunktionen. Neben der komfortablen Steuerung mithilfe der App auf dem Smartphone bleibt die manuelle Nutzung über den vorhandenen Schlüssel weiterhin jederzeit möglich — auch bei leeren Batterien oder einem Ausfall der App. Im folgenden Abschnitt werden die zentralen Funktionen und technischen Möglichkeiten des Schlosses im Detail beschrieben.

**Entsperren/Verriegeln** Das Entsperren und Verriegeln des Schlosses erfolgt über die Yale Home App, wobei ein Smartphone oder eine Smartwatch als digitaler Schlüssel dient. Die Steuerung der Schließmechanismen ist sowohl in unmittelbarer Nähe des Schlosses als auch aus der Ferne möglich. Für den Fernzugriff ist jedoch die Yale Connect Wi-Fi Bridge erforderlich, da sie die Funktion zur Fernsteuerung gewährleistet. [19], [21]

**Automatische Ent- und Verriegelung** Die Steuerung der Schließmechanismen kann auch automatisiert über die Funktionen Auto Unlock und Auto Lock des elektronischen Türschlosses erfolgen. Mithilfe der Geo-Fencing-Technologie erkennt das System, wenn sich das autorisierte Smartphone in einem zuvor definierten Radius zur Haustür befindet, und entriegelt in diesem Fall das Schloss automatisch. Beim Verlassen des Hauses sowie dem anschließenden Schließen der Tür wird das Schloss automatisch verriegelt. [19], [21] Die Zeitspanne bis zur Verriegelung kann nutzerspezifisch konfiguriert werden und liegt wahlweise zwischen sofortigem Verschluss und einer Verzögerung von bis zu 30 Minuten [18]. Auch hierfür wird die Yale Connect Wi-Fi-Bridge für den vollen Funktionsumfang benötigt.

**DoorSense™-Sensor** Die integrierte Türsensor-Technologie – DoorSense™ – ermöglicht die Erkennung des aktuellen Türzustands, also ob die Tür vollständig geschlossen ist oder offen steht. Bei einer nicht vollständig geschlossenen Tür wird der Eigentümer unmittelbar benachrichtigt, wodurch eine erhöhte Betriebssicherheit gewährleistet werden kann. [19], [21]

**Virtuelle Zutrittskontrolle** Über die Yale Home App lassen sich digitale Zugangsschlüssel mit individuell definierbaren Zeitfenstern erstellen und an autorisierte Personen — etwa Familienmitglieder, Mitbewohner oder temporäre Gäste — übertragen, um diesen kontrollierten Zutritt zum Gebäude zu ermöglichen [18], [21]. Die zugrunde liegenden Zugangskategorien umfassen dabei:

- einen dauerhaften Zugang ohne zeitliche Einschränkungen (z. B. für enge Angehörige),
- einen wiederkehrenden Zugang, der sich in regelmäßigen Intervallen aktivieren lässt (etwa für Besucher, die wöchentlich an einem bestimmten Wochentag und zu festgelegten Zeiten Zutritt erhalten),
- sowie einen zeitlich begrenzten Einmalzugang, der nach Ablauf des zuvor definierten Zeitrahmens automatisch seine Gültigkeit verliert [31].

Die Kontrolle über diese virtuellen Schlüssel verbleibt dabei jederzeit beim Eigentümer des Smart Locks: Berechtigungen können flexibel angepasst, deaktiviert oder vollständig entzogen werden [31]. Im Falle eines Verlusts oder Diebstahls eines autorisierten Smartphones bietet das System zusätzlich eine sicherheitsrelevante Schutzmaßnahme: Der entsprechende digitale Schlüssel kann unmittelbar über die App deaktiviert und bei Bedarf durch einen neuen, kostenfrei generierten Schlüssel ersetzt werden [18].

**Option „Telefon verloren“** Über die Internetadresse *https://account.aaecosystem.com/login* besteht die Möglichkeit, ein Mobilgerät als verloren zu melden. Es ergibt sich die Möglichkeit, sich mit der in der Anwendung hinterlegten E-Mail-Adresse zu authentifizieren, um das gestohlene oder verlorene Endgerät aus dem System zu entfernen. Dadurch wird verhindert, dass unbefugte Personen Zugriff auf sensible Daten oder Funktionen erhalten können. [32]

**Vernetzung mit Sprachassistenten** Das Smart Lock lässt sich zudem mit gängigen Sprachassistenten wie Google Assistant oder Amazon Alexa verknüpfen. Diese Vernetzung ermöglicht die sprachgesteuerte Bedienung des Schlosses, wodurch Zugangs- und Steuerungsfunktionen auch kontaktlos und effizient ausgeführt werden können. Für die Nutzung dieser Fernzugriffsmöglichkeiten ist jedoch die zusätzliche Einrichtung der Yale Connect Wi-Fi-Bridge erforderlich, welche die sichere Kommunikation zwischen dem Schloss und den Cloud-Diensten der Sprachassistenten gewährleistet. [18], [21]

**Ereignisprotokoll** Über die zugehörige Smartphone-App kann ein detailliertes Ereignisprotokoll abgerufen werden, das sämtliche Zugangsaktivitäten dokumentiert. Dieses Protokoll ermöglicht es dem Eigentümer, jederzeit und ortsunabhängig nachzuvollziehen, welche berechtigten Personen zu welchem Zeitpunkt das Gebäude betreten oder verlassen haben. Dadurch wird nicht nur die Transparenz über alle Zutrittsvorgänge erhöht, sondern zugleich eine kontinuierliche Überwachung der Zugangshistorie sichergestellt — auch dann, wenn sich der Eigentümer selbst nicht vor Ort befindet. [33]

**Smart Alerts** Sogenannte Smart Alerts informieren in Form automatisierter Benachrichtigungen über sicherheitsrelevante Ereignisse wie manuelle oder nutzerspezifische Verriegelungen, geöffnete Türen oder Auto-Lock-Aktivitäten. Diese werden im Aktivitätsprotokoll der App dokumentiert. Für den vollen Funktionsumfang ist die Verbindung zur Yale Connect Wi-Fi-Bridge erforderlich. [34]

**Integration in Smart Home Systeme** Durch die Integration des Yale Linus® Smart Locks in ein bestehendes Smart-Home-System, wie beispielsweise das Bosch Smart Home System, lässt sich das Schloss über die im Lieferumfang enthaltene Yale Connect Wi-Fi-Bridge direkt mit der Bosch Smart Home App verknüpfen. Infolgedessen kann das Schloss nicht nur zentral gesteuert, sondern auch in umfassende Automatisierungsprozesse eingebunden werden, was sowohl den Sicherheitskomfort als auch die Nutzerfreundlichkeit deutlich erhöht. Im Mittelpunkt dieser Automatisierungen steht die direkte Verknüpfung des Schlosszustands – also des Verriegelns oder Entriegelns – mit vordefinierten Systemreaktionen. So kann das Verriegeln der Tür automatisch die Aktivierung eines Alarmsystems oder eines „Haus verlassen"-Szenarios auslösen, während das Entriegeln etwa zur Deaktivierung der Sicherheitsfunktionen oder zur Anpassung von Licht- und Raumklima führen kann – jeweils abhängig von der gewählten Konfiguration. Darüber hinaus besteht die Möglichkeit, zeitgesteuerte Verriegelungen einzurichten, sodass sich das Türschloss zu definierten Tageszeiten automatisch verschließt. [18]

## 2.6 Architekturkonzepte

Um die Funktionsweise und das Zusammenspiel der einzelnen Komponenten des Yale Linus® Smart Locks nachvollziehen zu können, ist ein hinreichendes Verständnis der zugrunde liegenden Kommunikations- und Systemarchitektur erforderlich. Im Folgenden werden daher zentrale Aspekte dieser Kommunikationsstruktur systematisch behandelt: Zunächst wird die übergeordnete IoT-Architektur skizziert, die das Smart Lock als Teil eines vernetzten Gesamtsystems einordnet. Anschließend folgt eine detaillierte Betrachtung der spezifischen Systemarchitektur des Yale Linus® Smart Locks, welche die internen Strukturen und Abläufe innerhalb des Geräts sowie dessen Schnittstellen zu anderen Systemen beschreibt.

### 2.6.1 IoT-Architektur und Server-Client-Prinzip

Die IoT-Architektur bildet die Grundlage für das Verständnis der technischen Struktur und des Zusammenspiels der verschiedenen Systemkomponenten [35] im Kontext des Yale Linus® Smart Lock Systems. Eine zentrale Bedeutung kommt dabei dem Client-Server-Modell zu, das die Kommunikation zwischen Endgeräten — wie Smartphones oder Smartwatches –, dem Smart Lock selbst sowie den angebundenen cloudbasierten Diensten organisiert [36]. Im Rahmen dieses Modells übernimmt beispielsweise die Yale Home App auf dem Smartphone die Rolle des Clients, welcher Anfragen wie bspw. das Entriegeln der Tür an einen zentralen Server übermittelt [36]. Im vorliegenden Fall an die Cloud-Plattform von Yale. Da die Kommunikation nicht direkt zwischen der Smartphone-App und dem Schloss erfolgt, sondern, neben der Kommunikation zwischen Smartphone-App und Cloud, weitere Komponenten in der Übertragungskette zwischen App und Schloss beteiligt sind (siehe Abbildung 2.4), lässt

sich Folgendes festhalten: In der Kommunikation zwischen Cloud und Heimrouter behält die Cloud die Rolle des Servers, während der Heimrouter die Daten weiterleitet. Somit agiert die Bridge als Client. Im Abschnitt zwischen Bridge und Schloss übernimmt die Bridge die Serverrolle, während das Schloss als Client agiert. Die folgenden Funktionen beziehen sich auf den Cloud-Server, der unabhängig von der Rolle einzelner Geräte auf den Zwischenstrecken die zentrale Verarbeitung übernimmt [37]. Die Cloud verarbeitet anschließend diese Anfragen, authentifiziert den Nutzer und veranlasst daraufhin entsprechende Aktionen, etwa die Freigabe des Schlossmechanismus und damit das Öffnen der Tür [38]. Darüber hinaus kann der Server auch eigenständig Informationen oder Steuerbefehle an die Clients übermitteln, etwa in Form von Push-Benachrichtigungen bei Statusänderungen des Schlosses oder sicherheitsrelevanten Ereignissen [39]. Dies wird bspw. in Message Queuing Telemetry Transport-basierten IoT-Architekturen durch das Publish/Subscribe-Prinzip umgesetzt, bei dem der Server Nachrichten in abonnierten Topics veröffentlicht [39]. Das Modell ermöglicht eine skalierbare, zentral steuerbare Interaktion zwischen den Geräten und unterstützt sowohl lokale als auch entfernte Zugriffsmöglichkeiten. [36], [38]

Darauf aufbauend beschreibt die IoT-Architektur die strukturelle Einteilung des Systems in vier funktionale Schichten: die Sensor-, Netzwerk-, Verarbeitungs- und Anwendungsschicht. Diese Schichtenstruktur liefert einen konzeptionellen Rahmen zur systematischen Einordnung der Hardwarekomponenten, Kommunikationswege, Datenverarbeitungsprozesse und nutzerbezogenen Anwendungen, die im Zusammenspiel die Funktionalität des Smart Lock Systems gewährleisten. [36]

Der Aufbau der grundlegenden IoT-Architektur ist in Abbildung 2.2 einzusehen. Die konkrete Umsetzung der IoT-Architektur im Kontext des Yale Linus® Smart Locks ist dagegen in Abbildung 2.3 dargestellt. Hierbei werden die konkreten Schichten des Geräts den allgemeinen IoT-Schichten zugeordnet.

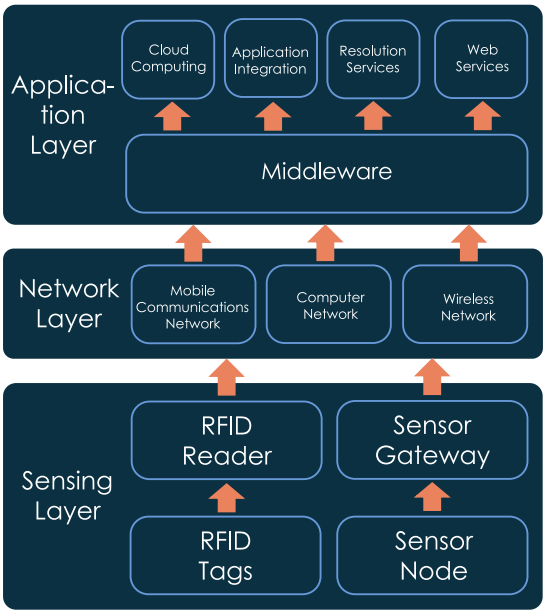



**Abbildung 2.2:** Grundlegende IoT-Architektur, basierend auf: [36]

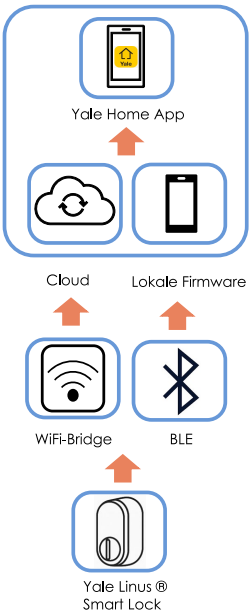



**Abbildung 2.3:** IoT-Architektur des Yale Linus® Smart Locks

Die Sensorschicht stellt die erste funktionale Ebene im IoT-System dar. Ihre zentrale Aufgabe besteht in der Informationserfassung im gesamten Netzwerk. Sie umfasst sowohl Sensoren als auch Aktoren, die physikalische Zustände und Veränderungen aus der realen Umgebung detektieren und in digitale Signale überführen. Beim Yale Linus® Smart Lock zählen hierzu unter anderem der Türzustandssensor, welcher erkennt, ob die Tür offen oder geschlossen ist, sowie der integrierte Motor, der den Schließmechanismus elektronisch steuert. [36]

Die Netzwerkschicht übernimmt die Aufgabe der Datenübertragung zwischen den physischen Komponenten des Systems — dem Smart Lock und der Yale Connect Wi-Fi-Bridge – sowie der Cloud-Infrastruktur. Sie sorgt für die sichere und zweckmäßige Weiterleitung der vom Sensorsystem erfassten Informationen an zentrale Verarbeitungseinheiten, etwa den Server bzw. cloudbasierte Dienste. In diesem Kontext kommen unterschiedliche Kommunikationsprotokolle zum Einsatz: BLE für die Verbindung zur mobilen Anwendungssoftware und Wi-Fi, welches über die Yale Connect Wi-Fi-Bridge bereitgestellt wird und dem Fernzugriff dient. Eine detaillierte Erläuterung dieser Technologien erfolgt in Unterabschnitt 2.7.1. Darüber hinaus gewährleistet diese Schicht die Aufrechterhaltung grundlegender Sicherheitsmerkmale wie Authentizität, Vertraulichkeit, Integrität und Verfügbarkeit der übertragenen Daten. [36]

Die Verarbeitungsschicht übernimmt die logische Kontrolle über die weitergeleiteten Datenströme und fungiert als Vermittlungsinstanz zwischen Netzwerk- und Anwendungsschicht. Sie verarbeitet eingehende Ereignisse und leitet daraus Aktionen ab. In der Yale-Systemarchitektur übernimmt das Cloud-Backend diese Rolle: Es führt Authentifizierungen durch, verarbeitet Ereignisse in Echtzeit und stellt sicher, dass Daten strukturiert und sicher gespeichert werden. [37]

Die Anwendungsschicht stellt die oberste Ebene der IoT-Architektur dar und ist direkt auf die Interaktion mit dem Endnutzer ausgerichtet. Über grafische Benutzeroberflächen — wie in dem speziellen Fall die Yale Access App — werden Funktionen wie das Öffnen oder Verriegeln der Tür, die Verwaltung digitaler Schlüssel oder das Einsehen von Aktivitätsprotokollen ermöglicht. Darüber hinaus dient sie als Plattform für Benachrichtigungen, Statusanzeigen und Automatisierungen. Die App agiert dabei als Client im Client-Server-Modell und erhält Informationen vom Backend, verarbeitet Nutzereingaben und initiiert Steuerbefehle an das Schlosssystem. [36]

### 2.6.2 Systemarchitektur des Yale Linus® Smart Schlosses

Aufbauend auf der zuvor beschriebenen IoT-Architektur folgt nun die konkrete Darstellung der Systemarchitektur des Yale Linus® Smart Locks. Im Fokus stehen dabei die technischen Kommunikationsbeziehungen zwischen den einzelnen Systemkomponenten. Dem Smart Lock selbst, der Yale Connect Wi-Fi-Bridge, der mobilen Anwendungssoftware sowie den cloudbasierten Backend-Systemen.

Die Systemarchitektur in Abbildung 2.4 zeigt auf, wie diese Elemente über verschiedene Schnittstellen und Protokolle miteinander interagieren, um eine sichere und zuverlässige Steuerung des Schlosses zu gewährleisten.

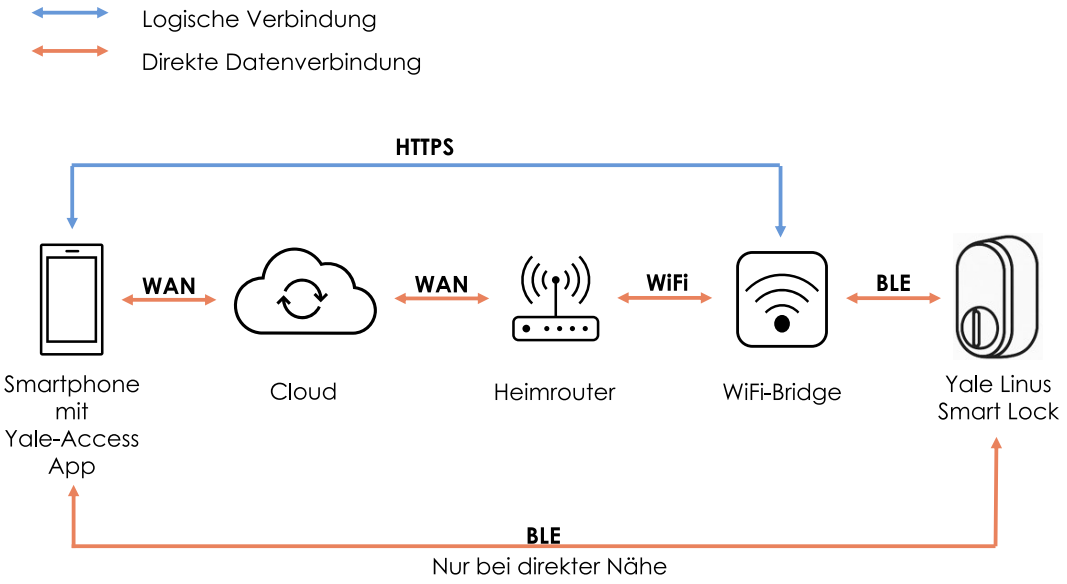



**Abbildung 2.4:** Systemarchitektur des Yale Linus Smart Lock – Kommunikationswege zwischen App, Cloud, Bridge und Smart Lock

Die Verbindung zwischen dem Smart Lock und der Yale Connect Wi-Fi-Bridge erfolgt über das energiesparende BLE-Protokoll (Version 4.2, $2.4\,\mathrm{GHz}$) [40]. Diese Verbindung ist ausschließlich auf kurze Distanzen ausgelegt und ermöglicht eine stabile, latenzarme Kommunikation zwischen Schloss und Bridge im lokalen Nahbereich [41]. Die Bridge dient in diesem Kontext als Vermittler zwischen dem lokal installierten Schloss und der cloudbasierten Infrastruktur von Yale [40].

Zur Übermittlung von Steuerbefehlen, Statusdaten oder Konfigurationsänderungen an das Cloud-Backend wird die Wi-Fi-Bridge über das lokale Heimnetzwerk (Wi-Fi 802.11 b/g/n, $2.4\,\mathrm{GHz}$) [42] mit dem Internet verbunden. Dabei fungiert der WLAN-Router als Multifunktionsgerät, das verschiedene Netzwerkkomponenten integriert: Die Daten passieren zunächst die Firewall OPNSense und werden anschließend über einen Switch an die Fritzbox weitergeleitet, welche als Access Point im lokalen Netzwerk agiert. Von dort aus stellt die Fritzbox die Verbindung zur Wi-Fi-Bridge her, welche die Kommunikation zwischen dem Smart Lock und den cloudbasierten Diensten ermöglicht. Für die Datenübertragung wird ein TCP/IP-basiertes Netzwerkprotokoll verwendet, das in Kombination mit TLS [40] die Vertraulichkeit, Integrität und Authentizität der übermittelten Daten sicherstellt [43]. Unterabschnitt 2.7.2 behandelt die eingesetzten Verschlüsselungsverfahren, TLS und AES-128-Bit, eingehender.

Die mobile Yale Access App interagiert je nach Nutzungsszenario auf zwei unterschiedlichen Wegen mit dem Schlosssystem: Im Nahbereich kommuniziert sie direkt per BLE mit dem Schloss, um dieses beispielsweise zu ver- oder entriegeln. Beim Fernzugriff — also außerhalb der Bluetooth-Reichweite — erfolgt die Kommunikation zwischen App und Schloss über eine verschlüsselte Internetverbindung. Die Cloud fungiert hierbei als Vermittler, während die Bridge eine Schnittstelle zur Verbindung mit dem Schloss darstellt. [40]

Das Zusammenspiel dieser Kommunikationspfade folgt ebenfalls dem Client-Server-Prinzip: Die mobile App agiert als Client, der Nutzereingaben aufnimmt und diese über definierte Schnittstellen an die Serverinfrastruktur übermittelt. Dort erfolgt die zentrale Verarbeitung,

inklusive Authentifizierung, Autorisierung sowie Steuerung des Schlosses. Ergänzend dazu werden Ereignisse wie Türöffnungen serverseitig erfasst und aktiv an den Client übermittelt, etwa per Push-Benachrichtigung. [44]

Wie aus Simeoni et al. [45] hervorgeht, schafft die Kombination aus lokaler BLE- Kommunikation, internetbasierter Fernsteuerung über eine Wi-Fi-Bridge und zentraler Datenverarbeitung in der Cloud ein hybrides System. Aufbauend darauf lässt sich ableiten, dass ein solches System sowohl lokal autonom agieren als auch global vernetzt operieren kann. Die resultierende Systemarchitektur vereint somit Sicherheitsanforderungen, Benutzerfreundlichkeit und Skalierbarkeit in einem integrierten Gesamtsystem.

## 2.7 Verwendete Technologien des Yale Linus® Smart Schlosses

Für eine effiziente und zuverlässige Kommunikation zwischen dem Smart Lock und der Benutzersteuerung sind verschiedene technische Komponenten und Protokolle erforderlich, die eine nahtlose Interaktion sicherstellen. Neben den Kommunikationsschnittstellen spielen auch die verwendeten Protokolle eine zentrale Rolle.

Im Folgenden werden daher die relevanten Kommunikationsschnittstellen BLE und Wi-Fi erläutert. Darauf aufbauend erfolgt eine Analyse der zugrundeliegenden Kommunikationsprotokolle auf Basis des TCP/IP-Modells sowie deren Einordnung in das Open Systems Interconnection-Referenzmodell, welches die Netzwerkschichten zur Strukturierung der Datenkommunikation beschreibt. Abschließend werden die Sicherheitsmechanismen analysiert, mit besonderem Fokus auf die eingesetzten Technologien AES-128-Bit zur Datenverschlüsselung und TLS zur Absicherung der Datenübertragung.

### 2.7.1 Kommunikationsschnittstellen

Wie in Unterabschnitt 2.6.2 bereits erläutert, nutzt das Yale Linus® Smart Lock verschiedene drahtlose Kommunikationstechnologien, um eine Verbindung zwischen dem Schloss, der Benutzeranwendung sowie den cloudbasierten Diensten herzustellen. Die Datenübertragung erfolgt dabei je nach Anwendungsszenario entweder lokal, etwa über das Smartphone des Nutzers, oder über das Internet mithilfe zusätzlicher Hardware wie der Yale Connect Wi-Fi-Bridge. Im Rahmen der folgenden Abschnitte werden zunächst die grundlegenden Funktionsweisen sowie die technischen Komponenten der beiden zentralen Übertragungsstandards — BLE und Wi-Fi — dargestellt.

**Bluetooth Low Energy** BLE ist ein drahtloses Kommunikationsprotokoll, das von der Special Interest Group (SIG) standardisiert wurde und seit der Einführung des Bluetooth 4.0-Standards im Jahr 2010 verfügbar ist. Es ergänzt die bestehende Bluetooth-Spezifikation gezielt für Anwendungen im Bereich des IoT, bei denen ein geringer Energieverbrauch eine zentrale Anforderung darstellt. Der Fokus liegt dabei auf der stromsparenden Kommunikation über kurze Distanzen. Diese Energieeffizienz wird unter anderem durch eine reduzierte Datenübertragungsrate von etwa $1\,\mathrm{Mbit/s}$ sowie die Verwendung der Personal Area Network-Technologie ermöglicht. Im Vergleich zu Bluetooth Classic zeichnet sich BLE durch deutlich geringere Anforderungen an Energie und Ressourcen aus, wodurch Geräte wie das Yale Linus®

Smart Lock, betrieben mit Batterien, über mehrere Wochen oder sogar Monate hinweg eigenständig betrieben werden können. Bereits ab Bluetooth-Version 4.2, wie sie im Yale Linus® Smart Lock implementiert ist, werden robuste Sicherheitsmechanismen eingesetzt, die eine zuverlässige Kommunikation gewährleisten. Hierzu zählt insbesondere der Advanced Encryption Standard im Counter-Modus mit Cipher Block Chaining Message Authentication Code (CCM), der zur Verschlüsselung der Nachrichten dient. Mit späteren Bluetooth-Versionen wurden BLE-Funktionalitäten um weitere Optionen wie höhere Reichweiten, Mesh-Netzwerke und variable Datenraten ergänzt. Da das Yale Linus® Smart Lock jedoch auf Bluetooth 4.2 basiert, sind solche erweiterten Funktionen, wie etwa ein dynamischer Wechsel zwischen verschiedenen Modi, bei diesem System nicht aktiv implementiert. [41]

**Aufbau BLE-Protokollstack** Der BLE-Protokoll-Stack ist modular aufgebaut und gliedert sich in drei zentrale Komponenten: Anwendung, Host und Controller. Eine schematische Darstellung der Architektur des BLE-Protokollstacks findet sich in Abbildung 2.5. Sie verdeutlicht die einzelnen Schichten und deren Zusammenwirken.

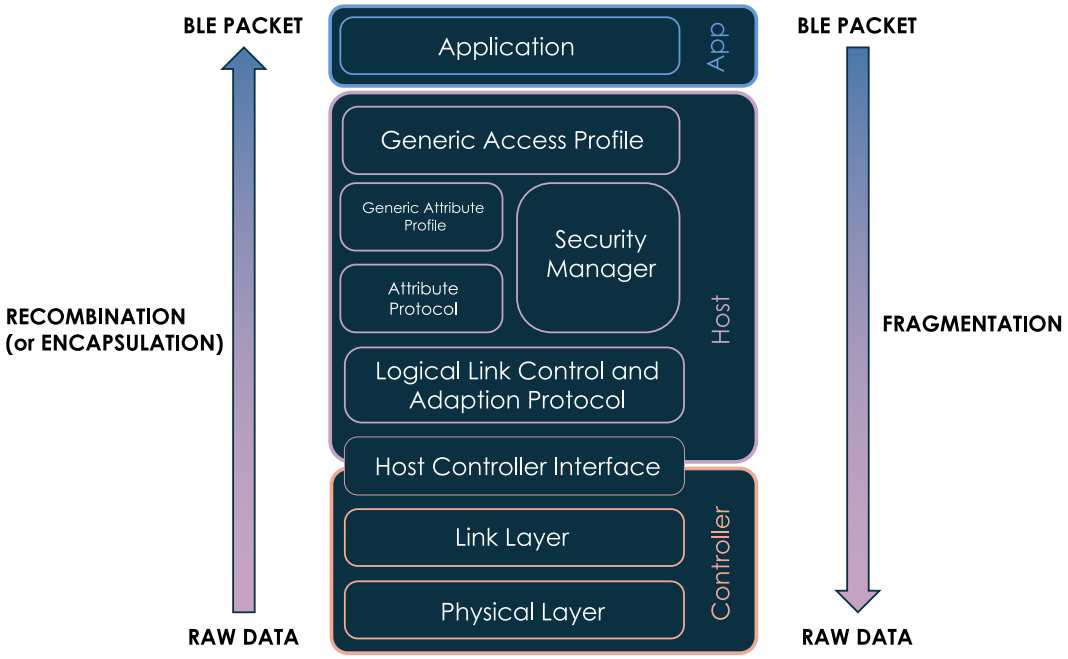



**Abbildung 2.5:** Bluetooth Low Energy Protokoll-Stack, basierend auf: [46]

Die oberste Ebene bildet die Anwendung, in der Regel eine mobile App, die als unmittelbare Schnittstelle zwischen dem Nutzer und dem IoT-System fungiert. Darunter befindet sich der Host, der verschiedene Protokollkomponenten umfasst, darunter das Generic Access Profile (GAP), das Generic Attribute Profile (GATT), das Attribute Protocol (ATT), das Security Manager Protocol (SMP) sowie das Logical Link Control and Adaptation Protocol (L2CAP). Diese Ebene ist für die logische Kommunikation und höhere Protokollfunktionen zuständig. Der Controller bildet die unterste Ebene des Stacks und besteht aus dem Link Layer (LL) sowie dem Physical Layer (PHY). Diese Komponenten sind direkt an der physikalischen Übertragung der Daten beteiligt. Die Kommunikation zwischen Host und Controller erfolgt über das standardisierte HCI, das als definierte Schnittstelle für den Informationsaustausch zwischen diesen beiden Funktionseinheiten dient. Insgesamt folgt der BLE-Stack einem vertikalen Schichtmodell: Datenpakete werden von der Anwendungsebene schrittweise durch die darunterliegenden

Protokolle verarbeitet, fragmentiert und schließlich auf physikalischer Ebene übertragen. Umgekehrt werden eingehende Datenpakete aus dem Physical Layer wieder zusammengesetzt und an die übergeordneten Ebenen weitergereicht, bis sie der Anwendung bereitstehen. [47]

Der PHY arbeitet im Frequenzbereich von $2.4\,\mathrm{GHz}$ innerhalb des Industrial, Scientific and Medical (ISM)-Bandes [46]. In diesem Modus erfolgt die Übertragung der Datenbits uncodiert im sogenannten 1M-Modus, wobei eine feste Datenrate von $1\,\mathrm{Mbit/s}$ realisiert wird [48]. Das verwendete Frequenzspektrum ist in insgesamt 40 Kanäle unterteilt, wobei die Kanäle 37, 38 und 39 speziell für die Übertragung von Werbepaketen reserviert sind, während die übrigen 37 Kanäle dem bidirektionalen Austausch von Datenpaketen innerhalb bestehender Verbindungen dienen [46].

Die Kommunikation zwischen den Geräten sowie deren Rollenzuweisungen innerhalb des Verbindungsaufbaus werden durch den LL gesteuert. Zu den dabei definierten Rollen zählen unter anderem der Advertiser und der Scanner [46]. Daraus abzuleiten ist, dass der LL für zentrale Prozesse verantwortlich ist: das Senden von Werbepaketen (Advertising), das Herstellen und Verwalten dauerhafter Verbindungen [46], das Frequenzsprungverfahren [48] sowie die Verschlüsselung mithilfe von AES [46].

Das HCI stellt die standardisierte Schnittstelle zwischen dem Controller und dem Host im BLE-Protokollstack dar. Es ermöglicht den Austausch von Befehlen, Ereignissen und Daten zwischen diesen beiden Komponenten. Dabei übersetzt es Rohdaten aus dem Physical Layer in strukturierte Datenpakete für den Host und umgekehrt. Da der BLE-Stack modular aufgebaut ist, werden Controller, Host und Anwendung getrennt implementiert. Das HCI ist somit essenziell, um eine einheitliche Kommunikation zwischen den logisch und physikalisch getrennten Ebenen zu gewährleisten. [46]

Das Logical Link Control and Adaptation Protocol übernimmt die Fragmentierung und Rekombination von Datenpaketen. Es zerlegt Daten aus höheren Schichten wie ATT und SMP in kleinere Einheiten für den LL und setzt empfangene Fragmente wiederum zu vollständigen Paketen für die oberen Schichten zusammen. Außerdem verwaltet diese Schicht das Multiplexing. [46]

Das Security Manager Protocol ist für die Ver- und Entschlüsselung der übertragenen Datenpakete verantwortlich und basiert auf einer Reihe definierter Sicherheitsalgorithmen [47] (AES und CCM) [41]. Darüber hinaus wirkt es sich auf sicherheitsrelevante Prozesse wie das Pairing, die Generierung von Schlüsseln und die Authentifizierung aus [47].

Im Attribute Protocol wird jedem Gerät eine Rolle entsprechend dem Client-Server-Modell zugewiesen [46]. Das Protokoll bildet die Grundlage für die Kommunikation innerhalb des GATT, indem es Daten in Form von Attributen strukturiert. Jedem Attribut sind dabei ein Handle, ein universell eindeutiger Bezeichner (Universally Unique Identifier (UUID)), spezifische Zugriffsrechte sowie ein zugehöriger Wert zugeordnet [46].

Die Schicht des Generic Attribute Profile definiert, wie strukturierte Daten in einer BLE- Verbindung zwischen zwei Geräten ausgetauscht werden [46]. Sie baut auf der darunterliegenden ATT-Schicht auf [46] und organisiert die zu übertragenden Informationen in einer hierarchi-

schen Struktur, bestehend aus Services (dt.: Diensten) und Characteristics (dt.: Merkmalen) [49] . GATT legt zudem die Kommunikationsrollen Client und Server fest und regelt den Zugriff auf Dienste und Merkmale, die der Server beim Aufbau einer Verbindung dem Client zur Verfügung stellt [46]. Ein Dienst stellt eine übergeordnete Einheit dar, die verwandte Attribute zusammenfasst, während die Merkmale die einzelnen Datenfelder innerhalb eines solchen Dienstes darstellen [46]. Jedes Merkmal beinhaltet einen Datenwert sowie optional zusätzliche Deskriptoren, die weiterführende Informationen über die Eigenschaft, deren Zugriffsrechte und Merkmale liefern [46]. Diese Eigenschaften bestimmen, welche Aktionen ein Client auf das jeweilige Merkmal ausführen darf [46]. Insgesamt ermöglicht die GATT-Schicht den strukturierten, dienstbasierten Zugriff auf spezifische Anwendungsdaten, wie etwa den Batteriestatus oder den Zustand einer Verriegelung [46].

Im Generic Access Profile werden sowohl die Geräterollen — etwa Advertiser und Scanner im Broadcasting-Modus sowie Master (Central) und Slave (Peripheral) im Connection-Modus — als auch die entsprechenden Modalitäten der Kommunikation, also Broadcasting und Connection, definiert. Darüber hinaus umfasst GAP die damit verbundenen Verfahren wie Advertising, Scanning und Connecting, welche den grundlegenden Ablauf für das Auffinden, Identifizieren und Verbinden von BLE-Geräten regeln. [50]

**Verbindungsaufbau unter BLE** Der Aufbau einer Verbindung zwischen zwei Geräten mittels Bluetooth Low Energy erfolgt in mehreren aufeinanderfolgenden Phasen. Zunächst initiiert das Yale Linus® Smart Lock als Advertiser und Peripheral den Prozess, indem es in regelmäßigen Abständen Advertising-Pakete [50] über die BLE-Werbekanäle 37, 38 und 39 sendet [46]. Ziel dieser Pakete ist es, die eigene Verfügbarkeit zu signalisieren und ein Central-Gerät — in diesem Fall ein Smartphone — zur Verbindungsaufnahme zu veranlassen [51].

Die Rollenverteilung sowie das Verhalten während des Verbindungsaufbaus sind durch das Generic Access Profile festgelegt, das u. a. bestimmt, dass das Schloss als connectable Advertiser (dt.: verbindbares Advertising-Gerät) agiert. [46], [51], [50]

Die physikalische Übertragung dieser Pakete auf den entsprechenden Frequenzen wird vom Physical Layer übernommen [46]. Die übergeordnete Steuerung des Verbindungsprozesses erfolgt durch den Link Layer [46], der unter anderem auch für die Anwendung des adaptiven Frequenzsprungverfahrens verantwortlich ist [48].

In der darauffolgenden Phase übernimmt das Smartphone sowohl die Rolle des Scanners als auch die des Central-Geräts. Es scannt die Umgebung regelmäßig, um empfangbare Advertising-Pakete zu identifizieren [51]. Sobald das Schloss erkannt wurde, wird es in der App des Nutzers angezeigt [52]. Anschließend sendet das Smartphone eine Verbindungsanfrage an das Schloss, welche vom Gerät angenommen wird und die erfolgreiche Verbindungsherstellung wird jeweils durch eine Nachricht an die Link Layer der beiden Geräte bestätigt [52]. Dadurch wird eine bidirektionale Kommunikationsbeziehung über BLE etabliert [52].

Für das Advertising muss ein spezifisches Advertising-Intervall konfiguriert werden, das den zeitlichen Abstand zwischen den ausgesendeten Werbepaketen definiert. Auf der Seite des Smartphones wiederum bestimmt das Scan-Intervall, in welchen Zeitfenstern das Funkmodul aktiv nach Werbesignalen sucht. Daraus lässt sich ableiten, dass das Schloss typischerweise

auf allen drei Advertising-Kanälen identische Pakete pro Intervall versendet, um die Wahrscheinlichkeit der Entdeckung durch den Scanner zu erhöhen. Zu beachten ist hierbei, dass das Smartphone als Scanner bei jedem Intervall den Kanal wechselt. [51]

Im Anschluss an die erfolgreiche Verbindung übernimmt das Schloss die Rolle des Slave, während das Smartphone als Master agiert. Zudem tauschen beide Geräte in regelmäßigen Abständen sogenannte Connection Events aus, welche die Grundlage für die Kommunikation auf den höheren Protokollschichten bilden. Die Verwaltung dieser Verbindung verbleibt beim Link Layer, während GAP weiterhin regelt, unter welchen Bedingungen eine Verbindung zulässig ist. [46]

Währenddessen erfolgt die Steuerung der Verbindung auf Hardwareebene über das Host Controller Interface, welche Befehle aus der App oder Firmware an den BLE-Controller übermittelt – etwa zum Starten von Advertising oder zur Annahme von Verbindungen. [46]

Sobald GATT-basierte Daten (z. B. ein „Unlock“-Befehl) übertragen werden, kommt das L2CAP zum Einsatz. Dieses Protokoll übernimmt die Fragmentierung größerer Datenpakete in BLE-konforme Einheiten, sorgt für die zuverlässige Übertragung und reassembliert die Datenpakete auf Empfangsseite. [46]

**Pairing unter BLE** Ein zentrales Sicherheitselement im BLE-Protokoll ist der Pairing-Prozess, bei dem zwei Geräte sowohl eine gegenseitige Authentifizierung als auch eine gemeinsame Schlüsselvereinbarung durchführen, um eine vertrauliche und manipulationssichere Verbindung aufzubauen [53]. BLE unterscheidet hierbei grundsätzlich zwischen zwei Pairing-Modi: Dem Legacy Pairing, wie es in den Spezifikationen 4.0 [53] und 4.1 [54] definiert ist, und dem ab Version 4.2 eingeführten Low Energy (LE) Secure Connections, das auf stärkerer Kryptografie basiert und eine verbesserte Widerstandsfähigkeit gegenüber Angriffen bietet [55].

Das Pairing-Verfahren in BLE gliedert sich unabhängig vom verwendeten Sicherheitsmodus in drei übergeordnete Phasen [54], [55]:

- Phase 1: Feature Exchange (dt.: Merkmalsaustausch)
- Phase 2: Authentication and Encryption (dt.: Authentifizierung und Verschlüsselung)
- Phase 3: Transport Specific Key Distribution (dt.: Übertragung transportspezifischer Schlüssel).

In Phase 1 tauschen die Geräte grundlegende Informationen über ihre Fähigkeiten aus, insbesondere hinsichtlich ihrer I/O Capabilities (dt.: Ein-/Ausgabefähigkeiten) und der unterstützten Sicherheitsfunktionen. Auf Basis dieser Informationen wird in Phase 2 ein geeignetes Verfahren zur Authentifizierung gewählt, das durch sogenannte Association Models (dt.: Zuordnungsmodelle) definiert ist. [54, 55]

Im Rahmen des Low Energy Legacy Pairing, wie es in der Bluetooth Core Specification 4.0/4.1 definiert ist, stehen drei Association Models zur Verfügung: Just Works, Passkey Entry und Out of Band (OOB). Das Modell Just Works kommt zur Anwendung, wenn ein oder beide Geräte nicht über geeignete Anzeige- oder Eingabemöglichkeiten verfügen. Es bietet kei-

ne nutzerbasierte Authentifizierung und ist somit anfällig für Man-in-the-Middle-Angriffe. Passkey Entry hingegen erlaubt eine gegenseitige Authentifizierung durch die Eingabe eines angezeigten sechsstelligen Zahlencodes auf dem jeweils anderen Gerät. Das OOB-Modell verwendet einen alternativen, zuvor etablierten sicheren Kanal, beispielsweise Near Field Communication (NFC), um einen gemeinsamen geheimen Wert auszutauschen. Abhängig vom gewählten Modell wird ein Temporary Key (TK) bestimmt, aus dem mithilfe einer kryptografischen Funktion ein Short Term Key zur Verschlüsselung abgeleitet wird. Nach erfolgreicher Verschlüsselung erfolgt in Phase 3 die gesicherte Übertragung weiterer Schlüssel wie des Long Term Key (LTK), des Identity Resolving Key oder des Connection Signature Resolving Key. [53], [54]

BLE Legacy Pairing übernimmt konzeptionelle Elemente des Secure Simple Pairing (SSP), das ursprünglich mit Bluetooth 2.1 für das klassische Bluetooth eingeführt wurde. Im Gegensatz zu SSP verzichtet BLE jedoch vollständig auf den Einsatz des FIPS-zugelassenen Elliptic Curve Diffie-Hellman (ECDH)-Verfahrens zur Schlüsselaushandlung sowie auf den Einsatz von HMAC-SHA-256 zur Sicherstellung von Authentizität und Nachrichtenintegrität. [53]

Mit der Einführung von Low Energy Secure Connections in der Bluetooth Core Specification 4.2 wurde das Pairing-Verfahren kryptografisch deutlich gestärkt, ohne jedoch die grundsätzliche Struktur des dreiphasigen Ablaufs zu verändern. Die wesentliche Erweiterung erfolgt in Phase 2, in der zusätzlich zu den bisherigen Association Models das Modell Numeric Comparison eingeführt wird. Dieses ist speziell für Geräte mit Anzeige- und Eingabemöglichkeiten vorgesehen und ermöglicht den Benutzern, eine auf beiden Seiten angezeigte sechsstellige Zahl visuell zu vergleichen und durch Eingabe zu bestätigen. Dadurch lässt sich die Authentizität der Gegenstelle prüfen und Man-in-the-Middle-Angriffe effektiv verhindern. [55]

Der Authentifizierungs- und Schlüsselvereinbarungsprozess basiert bei Low Energy Secure Connections auf dem kryptografischen Verfahren ECDH unter Verwendung der NIST P-256-Kurve. Jedes Gerät generiert ein eigenes Schlüsselpaar und tauscht mit der Gegenseite den öffentlichen Schlüssel aus. Anhand des empfangenen öffentlichen Schlüssels und des eigenen privaten Schlüssels berechnet jedes Gerät denselben geheimen Diffie-Hellman Key (DHKey). Dieser wird in der Folge durch die definierte Key-Derivation-Funktion $f5$ verarbeitet. Unter Einbeziehung weiterer Parameter wie Gerätezufallszahlen (Nonces) und Geräteadressen wird daraus deterministisch der LTK sowie ein MAC Key zur Authentizitätsprüfung abgeleitet. [55]

Der LTK dient in der dritten Phase der verschlüsselten Kommunikation und wird mit dem symmetrischen AES-CCM-Modus zur Absicherung der Verbindung eingesetzt. Ein wesentlicher Vorteil von Low Energy Secure Connections besteht darin, dass keine sensitiven Daten wie der TK, der Encrypted Diversifier oder die Zufallszahl (Rand) übermittelt oder gespeichert werden müssen. Diese Werte waren im Legacy Pairing erforderlich, um den passenden LTK bei erneuten Verbindungen wiederzuerkennen. In Secure Connections hingegen erfolgt die Schlüsselableitung vollständig auf Basis des gemeinsam berechneten DHKey. Dadurch wird das Sicherheitslevel gegenüber passivem Abhören wie auch gegen aktive Angriffe deutlich erhöht und die Datenintegrität verbessert. [55]

Da sich die exakte Pairing-Methode des verwendeten Yale Linus® Smart Lock nicht eindeutig aus der öffentlich zugänglichen Dokumentation ableiten ließ, wurde diese im Rahmen dieser Arbeit durch eine gezielte Analyse des BLE-Datenverkehrs ermittelt (siehe Abschnitt 4.1).

**Wireless Fidelity** Die Datenübertragung vom Schloss zur Bridge und umgedreht erfolgt analog zur Steuerung des Schlosses über ein Smartphone mittels BLE. Auf diese Weise lassen sich bei größerer räumlicher Distanz Informationen über die Cloud weiterleiten und das Schloss gezielt steuern, sodass gewünschte Aktionen ausgeführt werden können. Damit dieser Vorgang realisiert werden kann, greift die Bridge neben der BLE-Schnittstelle zusätzlich auf die Wi-Fi-Schnittstelle unter Verwendung des 802.11 b/g/n-Protokolls zurück. [42]

Wi-Fi basiert auf dem Standard des Institute of Electrical and Electronics Engineers (dt.: Institut der Elektro- und Elektronikingenieure) und ermöglicht eine drahtlose, leistungsfähige, verlässliche Kommunikation zwischen elektronischen Geräten sowie eine Anbindung dieser an kabelgebundene Netzwerke oder das Internet. Für die Verbindung mit dem Internet nutzt die Technologie dabei den TCP/IP-Stack. Die Technologie stellt unterschiedliche Betriebsmodi bereit, die insbesondere für IoT-Anwendungen vorteilhaft sind, da sie auf einen geringen Energiebedarf und wirtschaftliche Effizienz ausgelegt sind. Die Funkübertragung erfolgt primär in nicht lizenzierten Frequenzbereichen, vorrangig im 2,4-GHz-ISM-Band, kann jedoch auch im 5-GHz-Band betrieben oder als Dualband-Lösung genutzt werden, wodurch die Kosten der Datenübermittlung weiter reduziert werden. Trotz dieser Vorzüge unterliegt die Reichweite von Wi-Fi-Geräten gewissen Beschränkungen und die Qualität der Verbindung kann durch unterschiedliche Faktoren beeinträchtigt werden. [41]

Die Tabelle 2.4 gibt einen Überblick über die Protokolle, die von der Wi-Fi-Bridge verwendet werden. Dargestellt sind dabei insbesondere die zugehörigen Frequenzbereiche, Kanalbreiten sowie die maximal erzielbaren Datenraten.

**Tabelle 2.4:** Zusammenfassung der Wi-Fi-Protokolle [41]

| Protokoll | Frequenz | Kanalweite | Maximale Datenrate |
|---|---|---|---|
| 802.11 n | 2,4 oder 5 GHz | 20, 40 MHz | 450 $\mathrm{Mbit/s}$ |
| 802.11 g | 2,4 GHz | 20 MHz | 54 $\mathrm{Mbit/s}$ |
| 802.11 b | 2,4 GHz | 20 MHz | 11 $\mathrm{Mbit/s}$ |

### 2.7.2 Verschlüsselung und Sicherheitsprotokolle

**TCP-Handshake** Die Voraussetzung für den Aufbau einer gesicherten Verbindung mittels TLS ist zunächst die Herstellung einer Transportverbindung zwischen Client und Server. Diese erfolgt über den sogenannten TCP-3-Wege-Handshake: Zuerst sendet der Client ein Synchornize (SYN)-Paket, worauf der Server mit einem SYN/Acknowledgement (ACK) antwortet. Abschließend bestätigt der Client den Verbindungsaufbau mit einem ACK. Erst, nachdem dieser Ablauf abgeschlossen ist, kann der TLS-Handshake beginnen. [56]

**Transport Layer Security** TLS stellt anschließend einen geschützten Kommunikationskanal bereit und dient der Datenübertragung, also dem kontinuierlichen Übermitteln von Bytetströmen zwischen zwei Endpunkten. Grundlage für die Implementierung dieser Technologie ist

das Client-Server-Modell, das sich aus mehreren logischen Protokollkomponenten zusammensetzt: dem Handshake-, ChangeCipherSpec-, Alert- und Application Data-Protokoll. Alle diese Protokolle stützen sich auf die zentrale Komponente, das durchgehend aktive Record-Protokoll, das sowohl im ungesicherten als auch im chiffrierten Modus betrieben werden kann. Dieses baut auf dem TCP-Protokoll auf und nutzt dessen zuverlässige Übertragungseigenschaften. Der Record Layer nimmt Daten einer Anwendung entgegen und zerlegt diese in Records. Diese werden jeweils überprüft und gesichert. Die Verknüpfung der Records erfolgt über eine implizite Sequenznummer, die nicht mitgesendet wird. Zu Beginn wird für Absicherung und Validierung der Null-Algorithmus eingesetzt, was bedeutet, dass die Records zunächst ungeschützt sind. Jede Kommunikationsrichtung — vom Client zum Server und vom Server zum Client — verwendet dabei eigene Schlüssel und Sequenznummern, sodass zwei voneinander getrennte Datenströme entstehen. Die für die Kommunikation relevanten Algorithmen werden über das Handshake-Protokoll verhandelt. [57]

**TLS-Handshake** Der Ablauf des Handshakes lässt sich in drei wesentliche Schritte gliedern:

Vorgehensweise des TLS-Handshakes [57]

**Schritt 1:** ClientHello & ServerHello
Der Client teilt mit, welche TLS-Versionen und kryptografischen Verfahren er unterstützt. Der Server wählt daraus eine Ciphersuite, übermittelt seine Zufallszahl und sein Zertifikat und schließt den Nachrichtenblock mit *ServerHelloDone* ab.

**Schritt 2:** Schlüsselaustausch
Mithilfe des öffentlichen Schlüssels des Servers oder durch ein Diffie-Hellman-Verfahren wird ein gemeinsames Geheimnis (PremasterSecret) vereinbart. Daraus leiten beide Seiten das MasterSecret ab, das Grundlage für die Schlüsselableitung ist.

**Schritt 3:** Aktivierung der Sicherheit
Durch die ChangeCipherSpec-Nachricht wird der Wechsel in den gesicherten Modus angekündigt. Anschließend bestätigen Client und Server den erfolgreichen Aufbau mit der Finished-Nachricht.

Nach erfolgreichem Handshake übernimmt das Application Data-Protokoll die eigentliche Nutzdatenübertragung. Das Alert-Protokoll dient dem Austausch von Fehlermeldungen zwischen Client und Server und kann abhängig vom Verbindungszustand sowohl ungesichert als auch geschützt übertragen werden. [57]

**Advanced Encryption Standard** AES ist ein symmetrisches Blockchiffrierverfahren, das Daten in festen Blöcken von 128 Bit (16 Byte) verarbeitet [58]. Das Verfahren ver- und entschlüsselt Datenblöcke mit einem geheimen Schlüssel, wodurch es sich als Blockchiffre klassifiziert [58]. Der Algorithmus arbeitet auf einer 4x4-Byte-Matrix, dem sogenannten State AES, und verändert diese in mehreren iterierten Runden, um den Klartext in einen unlesbaren Chiffretext umzuwandeln [59]. Durch die byteorientierte Verarbeitung ist AES dabei in Hardware, als auch auf modernen Prozessoren, effizient einsetzbar [59].

Die Verschlüsselung beginnt mit einer initialen Schlüsseladdition, bei der der 128-Bit-Datenblock mit einem ersten Rundenschlüssel kombiniert wird. Danach durchläuft der State neun Runden, in denen die folgenden Schritte wiederholt werden. [58]

| Vorgehensweise des TLS-Handshakes [58] |
| --- |
| **Schritt 1:** Jedes Byte im Block durch ein anderes Byte ersetzen.<br>**Schritt 2:** Die Zeilen der Matrix zyklisch verschieben.<br>**Schritt 3:** Die Spalten der Matrix mischen.<br>**Schritt 4:** Den aktuellen Block erneut mit einem Rundenschlüssel kombinieren. |

Die letzte Runde verzichtet auf den dritten Schritt, das Mischen der Spalten, und besteht daher nur aus den Schritten eins, zwei und vier. Die Rundenschlüssel werden aus dem ursprünglichen 128-Bit-Schlüssel durch eine Schlüsselexpansion abgeleitet und in jeder Runde verwendet. AES-128 erzeugt durch diese Kombination von Austausch und Vermischung eine sichere Verschlüsselung, die den Klartext zuverlässig in Chiffretext überführt. [58]

# 3 Methodik

In diesem Kapitel wird das methodische Vorgehen zur Untersuchung der digitalen Spuren des Yale Linus® Smart Locks dargestellt. Ziel ist es, transparent aufzuzeigen, wie Netzwerk-, Cloud- und App-Daten systematisch erfasst, ausgewertet und im Hinblick auf ihre forensische Relevanz analysiert wurden.

Da ein Teil der Kommunikation verschlüsselt erfolgt, war die Auswahl geeigneter technischer Methoden — wie der Einsatz des Bluetooth-HCI-Snoop-Protokolls auf einem gerooteten Android-Gerät — essenziell, um überhaupt Zugriff auf aussagekräftige Daten zu erhalten. Ergänzend dazu wurden App-spezifische Daten über das Linux-Terminal extrahiert sowie in der Cloud gespeicherte Daten über eine Anfrage an den Anbieter eingeholt.

Das Kapitel gliedert sich in drei Abschnitte: Zunächst wird das gewählte Forschungsdesign sowie die allgemeine Herangehensweise erläutert. Im Anschluss daran folgen die verwendeten Erhebungsmethoden mit einem Fokus auf die technischen Maßnahmen zur Datenaufzeichnung und -extraktion. Abschließend wird die konkrete Durchführung der Datenerhebung, inklusive des Versuchsaufbaus, der eingesetzten Geräte und der manuellen Protokollierung, beschrieben.

## 3.1 Forschungsdesign und Herangehensweise

In diesem Abschnitt werden das Forschungsdesign sowie die grundlegenden Herangehensweisen zur Datenerhebung im Zusammenhang mit dem Yale Linus® Smart Lock beschrieben. Ziel der Untersuchung ist die systematische Analyse der Datenstrukturen und Kommunikationsprozesse, die im Zusammenhang mit der Nutzung des Schlosses entstehen. Der Fokus liegt dabei auf vier zentralen Datenquellen: den BLE-basierten Netzwerkverkehr, die App-Daten auf dem benutzten Smartphone sowie die Informationen, die in der Cloud gespeichert werden und dem Vergleich der Ereignisprotokolle.

Mit der zeitlich begrenzten experimentellen Datenerhebung soll der Netzwerkverkehr, insbesondere über BLE, betrachtet und nachvollzogen werden können. Ein wichtiges Ziel der Analyse bestand nicht nur darin, den unmittelbaren Datenverkehr während der Nutzung des Smart Locks zu erfassen, sondern auch herauszufinden, welche Informationen darüber hinaus dauerhaft in der App, auf dem Dateisystem des benutzten Smartphones und in der Cloud gespeichert werden. Dadurch ließ sich nachvollziehen, welche Daten auch nach Abschluss einer Aktion noch vorhanden sind und wie sich daraus Hinweise auf den zeitlichen Ablauf und die Art der ausgeführten Aktionen ableiten lassen. Ein weiterer Schwerpunkt liegt auf der Frage, inwiefern sich konkrete Nutzeraktionen mit den aufgezeichneten Systemereignissen in Beziehung setzen lassen. Zu diesem Zweck wurde ein Vergleich zwischen einem manuell erstellten Interaktionsprotokoll und dem automatisch durch die App generierten Ereignisprotokoll vorgenommen.

Zur Strukturierung der Analyse wurden folgende Forschungsfragen aufgestellt:

1) Wie lassen sich aus den während der Nutzung eines Yale Linus® Smart Lock erzeugten Netzwerk-, Cloud- und App-Daten digitale Spuren identifizieren, die forensisch relevant sind, und welchen Beitrag leisten diese zur Untersuchung sicherheitsrelevanter Vorfälle?
2) Welche Informationen lassen sich aus der zugehörigen Yale Home App extrahieren, über die die Aktionen gesteuert werden?

Zur Beantwortung der Forschungsfragen wurde eine qualitative, fallbasierte Analyse durchgeführt. Diese basiert auf einem konkreten Anwendungsfall: Der Untersuchung eines installierten Yale Linus® Smart Locks innerhalb eines festgelegten Untersuchungszeitraums. Das methodische Vorgehen ist technikgestützt und datenzentriert angelegt. Die Analyse verfolgt das Ziel, digitale Spuren zu identifizieren, die mit sicherheitsrelevanten Ereignissen in Verbindung stehen, und diese systematisch zu kategorisieren.

Die Datenerhebung umfasst:

- die gezielte Durchführung definierter Interaktionen mit dem Schloss über die mobile App
- die gleichzeitige Aufzeichnung des BLE-Traffics während dieser Interaktionen
- die Sicherung der App-Daten auf dem verwendeten Smartphone
- die Auswertung der in der Cloud gespeicherten Daten
- den Abgleich zwischen dem manuell erstellten Ablaufprotokoll und den automatisch generierten Einträgen in der App

Die Entscheidung für ein qualitatives Vorgehen basiert auf mehreren Aspekten. Es handelt sich um eine einzelfallbasierte, vertiefende Analyse eines spezifischen Systems, bei der keine statistische Verallgemeinerung im Vordergrund steht. Vielmehr wurde die Methodik gewählt, um aus technischer Perspektive ein möglichst umfassendes Bild über Struktur, Ablauf und Persistenz digitaler Spuren zu gewinnen. Das Vorgehen orientiert sich an den zuvor formulierten Forschungsfragen, bleibt dabei jedoch offen genug, um zusätzliche Erkenntnisse und Kategorien direkt aus den erhobenen Daten ableiten zu können. Klassifikationen und Strukturen wurden somit nicht ausschließlich im Vorfeld festgelegt, sondern entwickelten sich teils erst im Verlauf der Analyse.

In einzelnen Teilen der Analyse wurden ergänzend quantitative Elemente verwendet. Dazu zählen etwa die Zählung bestimmter BLE-Befehle, das strukturierte Erfassen von Zeitstempeln oder die wiederholte Ansprache spezifischer Attribute während bestimmter Aktionen. Diese Elemente dienen jedoch nicht der statistischen Auswertung im engeren Sinne, sondern der kontextbezogenen Einordnung technischer Abläufe und sind somit Bestandteil einer übergeordneten qualitativen forensischen Untersuchung.

Die Analyse wurde so angelegt, dass sie sich sowohl an etablierten technischen Standards — insbesondere im Bereich der IoT-Kommunikation und BLE — als auch an forensischen Anforderungen zur Nachvollziehbarkeit und Reproduzierbarkeit orientiert. Auf diese Weise

sollte gewährleistet werden, dass die Untersuchungsergebnisse fachlich fundiert und methodisch nachvollziehbar sind, um belastbare Aussagen zur digitalen Spurenerzeugung im Zusammenhang mit der Verwendung des Yale Linus® Smart Locks treffen zu können.

## 3.2 Erhebungsmethoden

Zur Untersuchung des Yale Linus® Smart Locks wurden verschiedene Datenquellen analysiert, um ein möglichst umfassendes Verständnis der Kommunikationsprozesse und Datenflüsse zu gewinnen. Die Datenerhebung erfolgte auf drei Ebenen: Netzwerk-, Cloud- und App-Daten. Diese Gliederung entspricht den zentralen Komponenten des Systems – von der lokalen Datenübertragung über die mobile Anwendung bis hin zur serverseitigen Infrastruktur.

Je nach Quelle kamen unterschiedliche technische Mittel zum Einsatz, etwa zur Aufzeichnung von Netzwerk- und Bluetooth-Kommunikation, zur Analyse der App-Daten über Universal Serial Bus (USB)-Debugging sowie zur Informationsbeschaffung über die Cloud-Dienste. Ergänzend dazu wurde eine manuelle Analyse durchgeführt, um die gewonnenen Daten mit dem in der App integrierten Ereignisprotokoll abzugleichen und so die Ergebnisse zu validieren und kontextualisieren.

Die folgenden Unterkapitel beschreiben im Detail die jeweils eingesetzten Hard- und Softwarekomponenten sowie das methodische Vorgehen zur Erhebung der Netzwerk-, Cloud- und App-Daten. Dabei wird erläutert, wie die jeweiligen Datenquellen erschlossen, welche technischen Rahmenbedingungen berücksichtigt und welche Herausforderungen bei der Datenerfassung überwunden wurden.

### 3.2.1 Netzwerk-Daten

Für die Netzwerkanalyse des Yale Linus® Smart Locks waren zwei Faktoren – zum einen der Mitschnitt des Datenverkehrs und zum anderen die Analyse dessen – von Bedeutung. Zur besseren Einordnung des methodischen Vorgehens erscheint zunächst eine kurze theoretische Einordnung in den Bereich der Netzwerkforensik sinnvoll.

Die Netzwerkforensik ist ein Teilbereich der digitalen Forensik, der sich mit der Sicherung, Analyse und Interpretation netzwerkbezogener Daten befasst. Ziel ist es, sicherheitsrelevante Vorfälle zu untersuchen, Angriffe zu identifizieren und digitale Beweise zu gewinnen, um verdächtige Aktivitäten nachvollziehen und gegebenenfalls strafrechtlich verfolgen zu können. Die Analyse stützt sich dabei auf verschiedene Methoden. Je nach Zielsetzung kommen beispielsweise Paketüberwachungen, Protokollanalysen oder spezialisierte Netzwerkforensik-Tools zum Einsatz. Beispiele dafür sind Wireshark oder tcpdump. Als Datenquellen dienen typischerweise Netzwerkkomponenten wie Router, Switches und drahtlose Access Points sowie Endgeräte wie Computer, Mobilgeräte oder Speichersysteme. [60]

Auf dieser Grundlage wurde der Netzwerkverkehr des Yale Linus® Smart Locks mit geeigneten Werkzeugen aufgezeichnet und analysiert. Im Folgenden wird das methodische Vorgehen zur Erhebung und Auswertung der Netzwerkdaten beschrieben.

Zur Aufzeichnung des Netzwerkverkehrs wurden zwei Methoden angewendet. Eine Methode bestand in der Verwendung der Open-Source-Software OPNSense, die auf einem Laptop installiert und als sogenanntes Monitoring-Gateway konfiguriert wurde. Dabei wurde der Datenverkehr gezielt über dieses Gateway geleitet, um alle ein- und ausgehenden Netzwerkpakete protokollieren zu können. OPNSense agierte in diesem Zusammenhang als transparenter Netzwerkbeobachter, ohne selbst aktiv in den Datenverkehr einzugreifen. Durch die Konfiguration entsprechender Filterregeln konnte der Datenverkehr gezielt nach der IP-Adresse des Smart Locks gefiltert werden, sodass ausschließlich relevante Datenpakete erfasst wurden. In der Benutzeroberfläche von OPNSense wurde hierfür ein Interface eingerichtet, dessen Konfiguration in Unterabschnitt 3.3.1 näher beschrieben wird. Die erfassten Daten wurden im `.pcap`-Format gespeichert und dienten als Grundlage für die nachfolgende Analyse. Da sich jedoch die Auswertung der erfassten Daten aufgrund der bei IoT-Geräten üblichen Verschlüsselung durch TLS als schwierig erwies, wurde ergänzend eine zweite Methode eingesetzt. Dabei wurde der BLE-Datenverkehr zwischen dem Smart Lock und dem Smartphone direkt mitgeschnitten. Dazu wurden das Smartphone und der Laptop über ein USB-C-Kabel verbunden und das Bluetooth-HCI-Snoop-Protokoll aktiviert. Es wurden mehrere Aktionen am Schloss durchgeführt und aufgezeichnet, um im Nachgang die Zuordnung einzelner Handles zu spezifischen Steuerfunktionen zu ermöglichen; die Mitschnitte wurden dabei im `.pcapng`-Format gespeichert. Die Speicherung der Dateien im `.pcap`- bzw. `.pcapng`-Format ermöglicht eine plattformübergreifende Weiterverarbeitung im Open-Source Netzwerk-Analysewerkzeug Wireshark [61]. Mit dessen grafischer Benutzeroberfläche lassen sich Netzwerkpakete gezielt untersuchen, indem spezifische Filter gesetzt werden. Darüber hinaus können die Protokollschichten des BLE-Protokollstacks visualisiert, deren Funktionen nachvollzogen und relevante Informationen extrahiert werden. [60]

Die beiden eingesetzten Methoden eröffneten jeweils spezifische Perspektiven auf die Kommunikationsabläufe zwischen Smart Lock, App und Infrastruktur. Ihre Kombination ermöglichte eine umfassendere Betrachtung und lieferte die Grundlage für die weiterführende Analyse.

### 3.2.2 Cloud-Daten

Die Erhebung der Cloud-Daten erfolgte durch eine direkte Anfrage bei Yale Home. Dazu wurde dem Support-Team des Unternehmens eine E-Mail über die in der App verwendete Gmail-Adresse gesendet. In dieser E-Mail wurde um Auskunft über die in der Cloud gespeicherten Daten gebeten. Diese sollten für eine nachfolgende Analyse verwendet werden. Die genaue Vorgehensweise wird in Unterabschnitt 3.3.2 beschrieben.

Da keine öffentlich zugänglichen Informationen darüber vorliegen, welche Daten das Unternehmen in der Cloud über seine Nutzer speichert, stellte die direkte Anfrage bei Yale Home die einzige Möglichkeit dar, verlässlich in Erfahrung bringen zu können, welche Daten tatsächlich auf den Servern des Unternehmens gespeichert sind. Dieses Vorgehen diente zugleich dem Abgleich mit den in der App gespeicherten Informationen.

### 3.2.3 App-Daten

Zur Erhebung App-spezifischer Daten kam die Android Debug Bridge (ADB) zum Einsatz. ADB ist ein Werkzeug, das von Google entwickelt wurde und für die Steuerung der Verwaltung sowie des Debuggings von Android-Geräten zuständig ist. Es basiert auf einer Client-Server-Architektur, die eine strukturierte Kommunikation zwischen den beteiligten Komponenten erlaubt. Der Austausch von Befehlen und Daten kann dabei entweder über eine TCP-Verbindung oder über eine USB-Schnittstelle erfolgen. [62]

Eine schematische Darstellung des Aufbaus der ADB-Struktur ist in Abbildung 3.1 zu sehen. Die Abbildung verdeutlicht die Beziehungen zwischen den Komponenten.

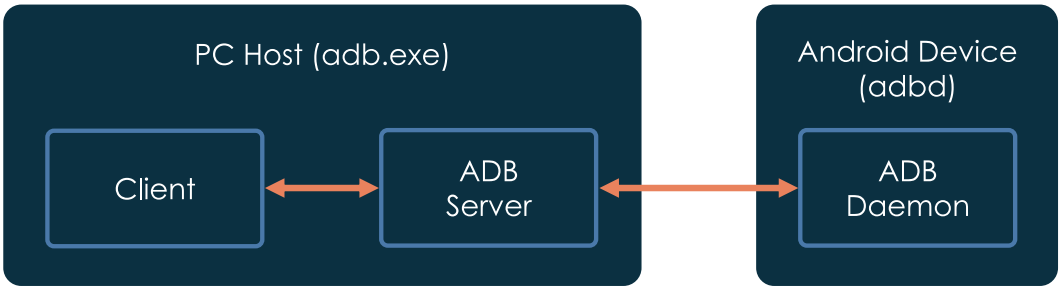



**Abbildung 3.1:** Aufbau der ADB-Struktur mit Fokus auf die drei wesentlichen Komponenten, basierend auf: [62]

In dieser Architektur übernimmt der Client die Rolle der Befehlsausführung durch den Nutzer über die Kommandozeile bzw. das Terminal des Host-Rechners. Der Server, der als Hintergrundprozess auf dem Host-System läuft, vermittelt zwischen dem Client und dem sogenannten Daemon (adbd). Dieser Daemon-Prozess ist auf dem Android-Gerät aktiv und empfängt die vom Server weitergeleiteten Kommandos, führt sie aus und übermittelt die Ergebnisse zurück. [62]

Im Rahmen der hier verwendeten Methodik wurde eine USB-Verbindung zur Kommunikation zwischen Laptop und Smartphone verwendet. In diesem Aufbau fungierte der auf dem Android-basierten Smartphone laufende Daemon als Zielinstanz, insbesondere im Kontext der installierten Yale Home App, über die das Yale Linus® Smart Lock gesteuert wird. Der Server-Prozess war als Hintergrunddienst auf dem Linux-basierten Laptop aktiv und übernahm die Vermittlung. Die eigentliche Interaktion mit dem System erfolgte durch manuelle Eingaben von ADB-Befehlen durch den Nutzer im Terminal, womit die Rolle des Clients erfüllt wurde. Der Einsatz von Terminal-Befehlen zur gezielten Extraktion und Analyse relevanter Log- und Konfigurationsdateien wird in Unterabschnitt 3.3.3 detailliert beschrieben.

Mithilfe der Android Debug Bridge lassen sich eine Vielzahl an datenbasierten Artefakten aus Android-Anwendungssoftwareen extrahieren, die im Rahmen einer forensisch orientierten Analyse von besonderer Relevanz sind — so auch im Fall der Yale Home App. Der Einsatz dieser Methode begründet sich durch die Möglichkeit, direkt auf App-interne Speicherbereiche zuzugreifen und dort gespeicherte Informationen systematisch zu erheben und zu analysieren.

Im Mittelpunkt der Datenextraktion steht das geschützte App-Verzeichnis unter `/data/data/<package-name>/`, in dem Android-Anwendungen sämtliche intern verwalteten Daten ablegen. Dieser Speicherbereich ist für das Betriebssystem reserviert und für andere Anwendungen standardmäßig unzugänglich. Für analytische Zwecke bietet er jedoch – bei entsprechender Berechtigung – Zugriff auf alle persistenten Dateien, die eine App im Rahmen ihrer Funktion erzeugt oder nutzt. [63]

Ein besonders relevanter Unterordner ist `/shared_prefs/`, in dem Konfigurations- und Zustandsdaten in Form von Extensible Markup Language (XML)-Dateien gespeichert werden. Diese beinhalten strukturierte Schlüssel-Wert-Paare, die typischerweise Informationen wie Benutzernamen, Anmeldestatus oder nutzerspezifische Einstellungen enthalten. Ihre lesbare Struktur erlaubt eine gezielte Extraktion einzelner Parameter, die zur Rekonstruktion der App-Nutzung herangezogen werden können. [63]

Weitere relevante Datenformate finden sich in relationalen Structured Query Language (SQL)-Datenbanken, die beispielsweise Nutzerverläufe oder protokollierte Interaktionen enthalten können. Diese Datenbanken liegen im Format `.db` oder `.sqlite` vor und sind oft mit transaktionsbezogenen Zusatzdateien wie `.journal` oder `.shm` verbunden, welche der Integritätssicherung während schreibintensiver Prozesse dienen. [63]

Temporär gespeicherte Inhalte — etwa Bilddateien, dynamisch geladene App-Inhalte oder Session-bezogene Daten — befinden sich in den Cache-Verzeichnissen der App. Diese Dateien werden vom System automatisch gelöscht, wenn der Speicherplatz knapp oder die App deinstalliert wird. [63]

Log-Dateien stellen eine weitere relevante Datenquelle dar. Diese enthalten meist laufzeitbezogene Meldungen, darunter Informationen zu App-Aktivitäten, Fehlerzuständen und Interaktionen mit Systemkomponenten. Die Daten werden nicht dauerhaft gespeichert, können jedoch während des App-Betriebs beobachtet und protokolliert werden. [63]

Ebenfalls Bestandteil des internen App-Verzeichnisses sind sogenannte Native Libraries im Format `.so`. Diese werden von der App zur Ausführung eingebunden und enthalten zentrale Komponenten der nativen Programmlogik. Sie können insbesondere bei der Analyse sicherheitsrelevanter Funktionen von Bedeutung sein. [63]

Schließlich lassen sich temporäre Zustands- und Sitzungsdaten identifizieren, die während der Laufzeit generiert und in verschiedenen Verzeichnissen, z. B. in Shared Preferences, im Cache oder in temporären Datenbankeinträgen, abgelegt werden. Diese enthalten Informationen über den aktuellen Nutzungskontext, zuletzt ausgeführte Aktionen oder Session-bezogene Zustände und erlauben somit eine Momentaufnahme des Anwendungsgeschehens. [63]

Die gezielte Extraktion dieser unterschiedlichen Datenformate und Speicherorte über ADB – unter Berücksichtigung gerätespezifischer Zugriffsvoraussetzungen wie Root-Rechten – stellt eine zentrale Komponente der hier angewandten Methodik dar und unterstreicht die forensische Eignung dieses Werkzeugs [63].

### 3.2.4 Manueller Vergleich der Ereignisprotokolle

Für den manuellen Abgleich der stattgefundenen Ereignisse bzw. der ausgeführten Aktionen wurde zunächst eine Protokollvorlage erstellt. Diese enthält allgemeine Protokollinformationen wie Datum, Uhrzeit, Ort, Versuchsleitung sowie die verwendeten Geräte und den dokumentierten Versuchsverlauf. Letzterer ist tabellarisch aufgebaut und umfasst die vier Spalten „Nr.“, „Aktion“, „Uhrzeit“ und „Anmerkung“. Die Spalte „Nr.“ ordnet für eine vereinfachte Auswertung jeder ausgeführten Aktion eine Nummer zu. In der Spalte „Aktion“ wurden vor Beginn des Tests alle geplanten Handlungsschritte, die durchgeführt werden sollen, systematisch und in chronologischer Reihenfolge dokumentiert. Die Uhrzeit, zu der die jeweilige Aktion ausgeführt wurde, ist in der dritten Spalte „Uhrzeit“ vermerkt. Die Spalte „Anmerkung“ enthält ergänzende Informationen, u. a. zum ausführenden Nutzer, zum Dateinamen der Datei, in der der BLE-Datenverkehr protokolliert wurde, sowie zu besonderen Vorkommnissen. Im Abschnitt Bemerkungen können zusätzliche, relevante Informationen zur Datenerhebung vermerkt werden. Die vollständige Protokollvorlage ist dem Anhang mit der Tabelle C.1 beigefügt.

Im Zeitraum der Datenerhebung wurden anschließend verschiedene Funktionen des Schlosses getestet und sämtliche durchgeführten Aktionen lückenlos mit dem entsprechenden Zeitstempel – jener Uhrzeit, die auf dem Smartphone des Hauptnutzers angezeigt wurde – dokumentiert. Abschließend wurden Screenshots des App-Protokolls angefertigt, um die Dokumentation zu vervollständigen und einen Vergleich mit dem selbst erstellten Protokoll zu ermöglichen. Unterabschnitt 3.3.4 enthält eine ausführliche Darstellung der Vorgehensweise.

Die Erstellung eines eigenen Protokolls dient der Analyse, inwieweit das Ereignisprotokoll der Yale Home App sämtliche Ereignisse erfasst und wie präzise diese Angaben letztlich sind. Das Ergebnis dieser Methode soll Rückschlüsse auf die Verlässlichkeit des App-Protokolls zulassen. Für Endnutzer wie auch für forensische Zwecke im Rahmen strafrechtlicher Ermittlungen kann bereits eine kleine Abweichung entscheidend sein und möglicherweise Aufschluss über die Nutzung des Schlosses durch eine bestimmte Person geben oder diese ausschließen.

## 3.3 Durchführung der Datenerhebung

Abbildung 3.2 zeigt den schematischen Aufbau des Testumfeldes und das Zusammenwirken aller Komponenten.

Der Router mit integrierter OPNSense-Firewall leitete die Daten, die über die Wide Area Network (WAN)-Schnittstelle aus dem Internet empfangen bzw. über die Cloud gesendet wurden, über einen Switch an die Fritzbox weiter. Diese fungierte in diesem Zusammenhang als Access Point und stellte die Verbindung zur Yale Connect Wi-Fi-Bridge her. Die Weiterleitung der Daten von der Firewall zum Switch und anschließend zur Fritzbox erfolgte über kabelgebundene Ethernet-Verbindungen. Die Bridge kommunizierte wiederum über BLE mit dem Yale Linus® Smart Lock. Auf der anderen Seite der Kommunikationskette befand sich das Smartphone mit der installierten und eingerichteten Yale Home App. Dieses stellte zum einen über das lokale Netzwerk, also den Access Point der Fritzbox, eine Verbindung zum Internet her und kommunizierte auf dem beschriebenen Weg mit dem Schloss. Dabei ist zu beachten,

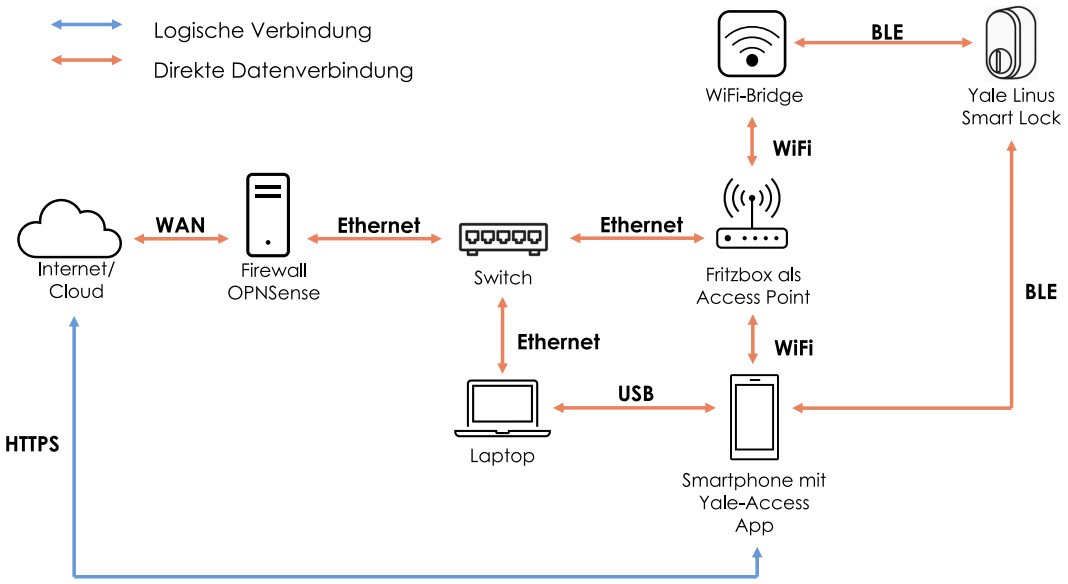



**Abbildung 3.2:** Erweiterte Systemarchitektur

dass sich sowohl die Bridge als auch das Smartphone zur Ersteinrichtung im selben Netzwerk befinden müssen. Zum anderen kann das Smartphone bei unmittelbarer Nähe zum Schloss auch direkt per BLE-Verbindung mit diesem kommunizieren.

Darüber hinaus war ein Laptop mit dem Betriebssystem Kali Linux über eine kabelgebundene Local Area Network (LAN)-Verbindung in das Testnetzwerk integriert. Das Smartphone war zusätzlich über eine USB-Schnittstelle physisch mit diesem Laptop verbunden, wodurch eine direkte Kommunikations- und Steuerungsmöglichkeit gegeben war.

Zum weiteren Versuchsaufbau ist Folgendes zu ergänzen: Zunächst wurde die Yale Home App auf dem gerooteten Smartphone installiert. Anschließend erfolgte die Erstellung eines Benutzerkontos sowie die Einrichtung der App und des Smart Locks gemäß Kapitel 2.4.2. Das Schloss wurde dabei nicht an einer Tür montiert, sondern auf einem Tisch positioniert. Dennoch konnten alle für die Datenerhebung relevanten Funktionen des Schlosses ausgeführt werden. Die DoorSense$^{TM}$-Technologie blieb in diesem Zusammenhang unberücksichtigt. Darüber hinaus wurde die Yale Home App ebenfalls auf einem nicht gerooteten Android-Smartphone, dem Samsung Galaxy S22, installiert. Dort wurde ein separates Benutzerkonto eingerichtet, das im weiteren Verlauf als Gastzugang diente. Dieses Vorgehen diente der Simulation unterschiedlicher Zugriffsrechte und Benutzerrollen im Rahmen der App-Nutzung. Detaillierte Informationen zu den verwendeten Smartphones sind in Tabelle 3.1 ersichtlich.

Die Datenerhebung erfolgte an einem einzelnen Tag innerhalb eines festgelegten Zeitraums. Die genauen Angaben hierzu sind der Tabelle C.1 im Anhang zu entnehmen.

### 3.3.1 Netzwerk-Daten

Für die Erhebung des Netzwerkverkehrs bildet sowohl das Verständnis des Kommunikationsablaufs als auch die zuvor beschriebene erweiterte Systemarchitektur die Grundlage der ersten Extraktionsmethode.

**Tabelle 3.1:** Verwendete Smartphones

| | **Google Pixel 7a** | **Samsung Galaxy S22** |
|---|---|---|
| Rolle | Hauptnutzer | Gastnutzer |
| Google-Konto | rxxxxxxxx.xxxxxxx@gmail.com | exxxxxxxxxxx@gmail.com |
| Telefonnummer | +49xxxxxxxxx3 | +49xxxxxxxxx |
| Android-Version | 14 | 15 |
| IP-Adresse | 10.0.1.112 | 192.168.178.31 |
| WLAN-MAC-Adresse | 94:45:xx:xx:xx:xx | 6C:AC:xx:xx:xx:xx |
| Bluetooth-Adresse | 94:45:xx:xx:xx:xx | 6C:AC:xx:xx:xx:xx |
| Yale Home App Version | 2025.8.0 | 2025.8.0 |
| Details | gerootet | nicht gerootet |

Dazu wurde in der OPNSense-Software über den Pfad `Interfaces > Diagnostics > Packet Capture` zu *Auftrag erstellen* navigiert und folgende Einstellungen vorgenommen: Das Interface wurde auf „igb1“ gesetzt, um sämtliche im System übertragenen Datenpakete zu erfassen. Sowohl die *Address Family* als auch das *Protocol* wurden auf „Any“ eingestellt, um IPv4- und IPv6-Adressen sowie sämtliche Protokolltypen, darunter bspw. TCP oder Internet Control Message Protocol, zu erfassen. Im Feld *Host Address* wurde die IP-Adresse (aus dem privaten reservierten IP-Adressbereich) der Bridge eingetragen, um ausschließlich die von diesem Gerät ein- und ausgehende Kommunikation zu erfassen. Das Feld *Count* definiert die maximale Anzahl der zu erfassenden Datenpakete. Standardmäßig wurde hier der Wert auf 100 Pakete eingestellt. Im Feld *Description* wurde der Name der jeweiligen Aktion vermerkt. Für die erste durchgeführte Aktion lautete dieser „Datenerhebung – Schloss entriegeln“. Die Aufzeichnung wurde durch Betätigung des „Play-Buttons“ gestartet. Unmittelbar danach erfolgte die Durchführung der ersten Aktion – das Entriegeln des Schlosses. Im Anschluss daran wurde die Aufzeichnung durch einen Klick auf den „Stop-Button“ beendet. Die erzeugte Datei wurde heruntergeladen und als ZIP-Archiv gespeichert. Dieses Archiv enthält eine JavaScript Object Notation (JSON)-Datei, welche die getätigten Einstellungen des Auftrags beinhaltet, sowie eine Packet Capture (PCAP)-Datei, die für die anschließende Analyse in Wireshark verwendet wurde.

Für die Analyse der mithilfe der OPNSense-Software erhobenen Netzwerkdaten wurden die erstellten PCAP-Dateien in Wireshark importiert und systematisch ausgewertet. Zunächst erfolgte über die Funktion *Statistiken* > *Endpunkte* eine vollständige Auflistung sämtlicher an der Kommunikation beteiligten IP-Adressen. Um unbekannte IP-Adressen identifizieren und zuordnen zu können, wurde ergänzend die Website *https://who.is/* verwendet.

Zur gezielten Analyse der Inhalte wurden unterschiedliche Filter in Wireshark eingesetzt: Der Filter `http` zeigt diejenigen Pakete an, die unverschlüsselt im Klartext übertragen werden, und liefert somit Informationen über den Sicherheitsgrad der Kommunikation. Darüber hinaus ermöglicht der Filter `tcp or tls` die Sichtbarkeit sämtlicher TCP- und TLS-Pakete. Dieser Filter dient der Überprüfung des Vorhandenseins verschlüsselter Elemente und stellt damit das funktionale Gegenstück zum Hypertext Transfer Protocol -Filter dar.

Im Rahmen der zweiten Extraktionsmethode wird in der nachfolgenden Schritt-für-Schritt-Anleitung die genaue Vorgehensweise für die Aufzeichnung des BLE-Datenverkehrs beschrieben.

| Aufzeichnug des BLE-Datenverkehrs | |
|---|---|
| **Schritt 1:** | Aktivierung des Bluetooth-HCI-Snoop-Protokolls in den Entwickleroptionen des Smartphones vom Hauptnutzer. |
| **Schritt 2:** | Für einen neuen Verbindungsaufbau, Bluetooth auf dem Gerät des Hauptnutzers aus- und wieder einschalten. |
| **Schritt 3:** | Eine Aktion in der Yale Home App ausführen. |
| **Schritt 4:** | Kurz warten, bevor die Log-Datei gezogen wird, um sicherzustellen, dass alle Daten erfasst wurden. |
| **Schritt 5:** | Extraktion der Log-Datei mittels ADB-Befehl. `adb shell „su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log“ >` *Dateiname* |
| **Schritt 6:** | Deaktivierung des Bluetooth-HCI-Snoop-Protokolls in den Entwickleroptionen des Smartphones vom Hauptnutzer. |

Der zuvor beschriebene Ablauf wurde für jede durch den Hauptnutzer ausgeführte Aktion wiederholt, um eine eindeutige Trennung der erfassten Daten für die spätere Analyse zu gewährleisten. Von dieser Vorgehensweise ausgenommene Aktionen sind das Ent- und Verriegeln des Schlosses aus der Entfernung durch den Hauptnutzer und alle Aktionen, die durch den Gastnutzer ausgeführt wurden. Das bewusste Durchführen mehrerer verschiedener Aktionen diente dazu herauszufinden, welches Handle für die jeweils durchgeführte Aktion angesteuert wird und somit einen Vergleich zwischen den verschiedenen Aktionen zu ermöglichen. Zur Übersicht sind die im Linux-Terminal ausgeführten Befehle in Abbildung A.12 dargestellt.

Nach Beendigung der Tests wurden die aufgezeichneten Log-Dateien zur Analyse in das Wireshark-Tool importiert, um eine detaillierte Untersuchung des BLE-Datenverkehrs sowie des zugrundeliegenden BLE-Protokollstacks vorzunehmen. Für die gezielte Auswertung wurden spezifische Wireshark-Filter eingesetzt und anderweitige Vorkehrungen getroffen, damit relevante Protokollinformationen selektiert werden konnten. Diese werden nachfolgend erklärt.

Über den Filter `hci_h4` werden sämtliche HCI-Pakete mit dem Format H4 angezeigt. Auf dieser Ebene erfolgt die Kommunikation zwischen Host und Controller. Mithilfe des genannten Befehls lässt sich diese Kommunikation detaillierter untersuchen.

Durch den Einsatz des Filters `btatt` werden ausschließlich ATT-Pakete sichtbar gemacht, die während des BLE-Datenverkehrs ausgetauscht wurden. Alle anderen Protokolle werden zur besseren Übersichtlichkeit sowie zur Reduktion von Rauschen ausgeblendet. Auf diese Weise können sämtliche Operationen, die auf der Ebene des ATT-Protokolls durchgeführt wurden und der Umsetzung einer Aktion dienen, nachvollzogen werden. Darüber hinaus ist es möglich, die Richtung der übertragenen Pakete sowie deren Quelle und Ziel zu bestimmen. Über das Dropdown-Menü des *Bluetooth HCI Access Control List*-Packet können hierzu weiterführende Informationen eingesehen werden.

In der Bluetooth Core Specification 4.2 sind sämtliche im BLE-Stack verwendeten ATT-Operationen definiert [55]. Dadurch ist es möglich, den jeweiligen Opcode dieser Operationen zu erfassen und anschließend als Filterkriterium zur Identifikation relevanter Informationen zu verwenden.

Zur Überprüfung, ob der Opcode mit den erzielten Ergebnissen übereinstimmt, wurde in der Paketliste eine zusätzliche Spalte angelegt. Dies erfolgte über einen Rechtsklick auf die Spaltenüberschrift, gefolgt von der Auswahl der Spalteneigenschaften. Anschließend wurde über das Plus-Symbol eine neue Spalte hinzugefügt. Als „Titel“, „Typ“ und „Feldname“ wurden dabei „Opcode (hex)“, „Custom“ sowie „btatt.opcode“ eingetragen. In der Spalte „Aufgelöst“ wurde das Häkchen entfernt und die Einstellungen mit „OK“ bestätigt. Daraufhin erschien die neu erstellte Spalte, in welcher der Opcode jeder ATT-Operation im Hex-Format angezeigt wird. Auf dieser Grundlage konnten die folgenden Analyseschritte durchgeführt werden.

Zur Ermittlung der Anzahl der genutzten Services wurde der Filterbefehl `btatt.opcode == 0x11` eingesetzt. Über das Dropdown-Menü des Bluetooth Attribute Protocol ließen sich anschließend für jeden Service der Name, der Start- und End-Handle sowie die jeweilige UUID dokumentieren.

Für die Erfassung der Charakteristiken wurde analog vorgegangen, wobei erneut der Filter `btatt.opcode ==` genutzt, der Opcode jedoch durch `0x09` ersetzt wurde. Dadurch konnten jeweils der Name, der Start-Handle, die zugehörige Service-UUID, der Characteristic Value Handle, die UUID der Charakteristik sowie die Characteristic Properties festgehalten werden. Über die Service-UUID, die jeder Charakteristik zugeordnet ist, war es möglich, eine Übersicht zu erstellen, welche die Zuordnung der Charakteristiken zu den entsprechenden Services darstellt.

Diese Übersicht konnte im Anschluss wie folgt abgeglichen werden: Über die obere Menüleiste in Wireshark lässt sich über *Wireless* > *Bluetooth ATT Server Attribute* eine Übersicht sämtlicher Services und deren zugehöriger Charakteristiken aufrufen. Dort sind zudem alle Start-Handles, Namen und UUIDs aufgelistet.

Zur Zuordnung von Deskriptoren und einem Client Characteristic Configuration Descriptor (CCCD) zu ihrer jeweiligen Characteristic wurde der Filter `btatt.opcode == 0x05` eingesetzt. Mit diesem Filter werden sämtliche *Find Information Response*-Pakete angezeigt, die im Rahmen der ATT-Kommunikation übertragen werden, und die zugehörigen Handles sowie UUIDs der jeweiligen Deskriptoren beziehungsweise CCCDs bereitgestellt.

Schließlich wurde zur Auswertung einzelner Werte der Filter `btatt` verwendet, ergänzt um den regulären Ausdruck *Read Response* in der Suchzeile. Dadurch war es möglich, den Wert einer Charakteristik auszulesen.

Zur Steigerung der Übersichtlichkeit der Daten wurde, analog zum Abgleich der Opcodes, in der Paketliste eine zusätzliche Spalte für die angesprochenen Handles sowie die darin enthaltenen Values erstellt. Die Vorgehensweise entspricht dabei derjenigen bei der Erstellung

der Opcode-Spalte. Die Inhalte der Spalteneigenschaften „Titel“, „Typ“ und „Feldname“ für die Handles und Values lauten jeweils „Handle“, „Custom“ und „btatt.handle“ sowie „Value“, „Custom“ und „btatt.value“.

Mit dem Filter `btatt.opcode == 0x12 || btatt.opcode == 0x52` werden alle Write Requests und Write Commands ausgegeben, die für die Steuerung der Aktionen von zentraler Bedeutung sind. Dadurch lassen sich die im Rahmen der Steuerung einer Aktion angesprochenen Handles identifizieren und die Bedeutung der gesetzten Werte nachvollziehen. Zudem ist es möglich, den Status eines CCCDs einzusehen.

### 3.3.2 Cloud-Daten

Für die Auswertung der in der Cloud gespeicherten Daten war es erforderlich, über die E-Mail-Adresse <*support@yalehome.de*> eine Anfrage an das Support-Team von Yale zu richten, um Auskunft über die dort gespeicherten Daten zu erhalten. Aus der Rückmeldung des Support-Teams ging hervor, dass Kunden ihre in der Cloud hinterlegten Aktivitäten über die Internet-Adresse *https://account.aaecosystem.com/login* mit ihren Yale-Home-Zugangsdaten herunterladen können.

Im Abschnitt „My Data“ befindet sich der Unterpunkt „Download my data“. Durch Betätigen des Buttons „Create a New Copy“ wird eine entsprechende Datei generiert. Nach Abschluss dieses Vorgangs erfolgt eine Benachrichtigung per E-Mail. Über das Cloud-Symbol im Untermenü „Click the cloud“ ist die erstellte Datei anschließend zum Herunterladen verfügbar.

Die heruntergeladene Datei liegt im Comma Separated Values (CSV)-Format vor. Yale weist ausdrücklich darauf hin, dass die Bereitstellung der Daten derzeit ausschließlich in diesem Dateiformat möglich ist. Diese Datei bildet somit die Grundlage für die Auswertung der in der Cloud gespeicherten Daten.

### 3.3.3 App-Daten

Eine der zentralen technischen Voraussetzungen für die erfolgreiche Erhebung der App-spezifischen Daten war der Root-Zugriff auf das Android-basierte Smartphone Google Pixel 7a. Der Superuser-Zugriff bildete die Grundlage für einen erweiterten Zugriff auf das Betriebssystem des Geräts. Dadurch war es möglich, auf Systemdaten der App, bspw. Logs oder Datenbanken, zuzugreifen und diese für die Analyse zu extrahieren. Für die Extraktion der relevanten Daten vom Smartphone wurde die Android Debug Bridge verwendet. Dieses Tool musste zuvor auf dem verwendeten Laptop installiert werden. Zur Nutzung dessen wurden Laptop und Smartphone per USB-Kabel (USB 3.0 oder USB-C) miteinander verbunden, wobei im Smartphone zuvor in den Entwickleroptionen das USB-Debugging aktiviert werden musste. Das USB-Debugging ermöglichte den Datenaustausch zwischen Laptop und Smartphone. Der für die Extraktion verwendete Laptop basierte auf einem Linux-Betriebssystem.

Zur verdeutlichten Darstellung, wie die App-spezifischen Daten extrahiert wurden, werden nachfolgend die verwendeten ADB-Befehle genauer erklärt. Die Befehle sind ebenfalls in der Abbildung A.13 und Abbildung A.14 zu sehen.

Über `adb shell pm list packages | grep yale` wird eine Shell geöffnet, die alle auf dem Gerät installierten Paketnamen listet. Diese Paketnamen wiederum werden nach dem Namen „yale“ gefiltert und alle gefundenen Pakete anschließend auf der Kommandozeile ausgegeben. Der `abd shell`-Befehl verbindet den Laptop über die ADB-Schnittstelle mit dem Android-Gerät des Hauptnutzers. Mit Hilfe von `su` werden Superuser-Rechte erreicht, mit denen auf geschützte Bereiche der App zugegriffen werden kann. Durch `cd /data/data/Verzeichnisname` wird ermöglicht, in das Verzeichnis zu wechseln, in dem die App-spezifischen Daten liegen (hier: `com.assaabloy.yale`). Der Befehl `ls` listet alle Inhalte des derzeitigen Verzeichnisses auf. Der `cp -r . /sdcard/Dateiname`-Befehl kopiert rekursiv, d. h. es werden sowohl alle Unterverzeichnisse als auch alle Dateien des aktuellen Verzeichnisses auf die öffentlich zugängliche SD-Karte des Gerätes in den neuen Ordner (hier: `backup-yale-linus`) kopiert. Durch das zweimalige Benutzen von `exit` wird die Sitzung, welche mit `adb shell` und `su` gestartet wurde, verlassen. Zum Schluss ermöglicht `adb pull /sdcard/backup-yale-linus` die Dateien von der SD-Karte auf den Laptop zu kopieren. Für die Analyse der erhobenen Daten kamen das Softwaretool Editor und der SQLite Browser zum Einsatz.

### 3.3.4 Manueller Vergleich der Ereignisprotokolle

Für den manuellen Vergleich zwischen dem Ereignisprotokoll der Yale Home App und dem selbst erstellten Protokoll wurden alle Zeitstempel der ausgeführten Aktionen in der Protokollvorlage erfasst. Dies ermöglichte eine nachträgliche Gegenüberstellung der eigenen Aufzeichnungen mit den App-Einträgen zur Überprüfung von Korrelationen zwischen Nutzeraktionen und aktionsbezogener Systemprotokollierung. Im Test wurden mithilfe der Yale Home App und des Schlosses folgende Funktionen durchgeführt und geprüft:

- Ent- und Verriegeln (einzeln)
- Ent- und Verriegeln (mehrfach hintereinander)
- Erteilen von Zugangsberechtigungen (wiederkehrend, temporär, immer, Eigentümer)
- Erstellen eines Smart Alerts („Gerät wird von einem bestimmten Benutzer ent- oder verriegelt.“)
- Auto Lock-Funktion (Verriegeln nach zehn Sekunden)
- Auto Unlock Funktion
- Ent- und Verriegeln des Schlosses aus der Ferne
- Einsehen des Ereignisprotokolls
- Ent- und Verriegeln durch den Gastnutzer

Für das Erteilen der Zugangsberechtigungen wurden sowohl der Vor- und Nachname als auch die Telefonnummer des Gastnutzers angegeben, um die Einladung versenden zu können. Nach Erteilen der ersten Zugangsberechtigung erfolgte der Login über das Google-Konte des Gastnutzers.

Im Anschluss an die Protokollierung aller ausgeführten Funktionen wurde innerhalb der Benutzeroberfläche der App zum Ereignisprotokoll navigiert und die Uhrzeit des Startzeitpunkts gesucht. Ab diesem Zeitpunkt wurden von sämtlichen im App-Protokoll vermerkten Aktionen Screenshots erstellt, um eine lückenlose Dokumentation zu gewährleisten. Auf dieser Grundlage war es möglich, den manuellen Vergleich vorzunehmen.

# 4 Ergebnisse

Im folgenden Kapitel erfolgt die Darstellung und Analyse der Ergebnisse, die aus der Anwendung der zuvor beschriebenen methodischen Vorgehensweisen hervorgegangen sind. Im Mittelpunkt stehen dabei die Untersuchung der Netzwerk-, Cloud- und App-Daten sowie der manuelle Vergleich der Ereignisprotokolle. Ziel dieser Ergebnisdarstellung ist es, die aus den einzelnen Analysen gewonnenen Informationen systematisch aufzubereiten und ihre Bedeutung im Kontext der beiden Forschungsfragen herauszuarbeiten.

Die Präsentation der Ergebnisse orientiert sich an den vier zugrundeliegenden Datenquellen: Zunächst werden die Resultate der Netzwerkanalyse vorgestellt, gefolgt von den Erkenntnissen aus den App-spezifischen und den Cloud-Daten. Abschließend erfolgt der Vergleich der Ereignisprotokolle, der sich auf die Dokumentation der in der App aufgezeichneten Ereignisse sowie auf das selbst erstellte Protokoll bezieht. Auf diese Weise können Gemeinsamkeiten und Unterschiede erkannt sowie detailliertere Beschreibungen festgestellt werden. Durch die strukturierte Aufbereitung wird eine klare Verbindung zwischen den eingesetzten Methoden und den daraus abgeleiteten Ergebnissen hergestellt und zugleich die Nachvollziehbarkeit des Forschungsprozesses gewährleistet.

## 4.1 Untersuchung der Netzwerk-Daten

Die Auswertung der in Wireshark importierten PCAP-Dateien, welche zuvor mit der OPNSense-Software erhoben wurden, ergaben folgende Beobachtungen: Nach der Identifikation und Zuordnung der IP-Adressen konnte festgestellt werden, dass die Adresse 10.0.1.138 der Yale Connect Wi-Fi Bridge zugeordnet werden konnte. Die beiden weiteren IP-Adressen 3.33.245.2 und 15.197.218.233 ließen sich Amazon Technologies Inc. zuordnen. Die analysierte Kommunikation erfolgte ausschließlich zwischen der Bridge-IP 10.0.1.138 und den beiden Cloud-IP-Adressen, d. h. von 3.33.245.2 zu 10.0.1.138 bzw. von 15.197.218.233 zu 10.0.1.138. In dem Zusammenhang wurden die zwei MAC-Adressen 70:61:be:c0:13:a7 und f4:90:ea:00:ff:f5 ersichtlich. Zwischen den beiden Cloud-IP-Adressen selbst konnte kein Verkehr festgestellt werden. Die IP-Adresse des Smartphones ist im aufgezeichneten Datenverkehr nicht enthalten. Die Anwendung der Filter `http` sowie `tcp or tls` zeigt, dass im Mitschnitt ausschließlich TCP- und TLS-Pakete versendet wurden. Weiterhin ließen sich ausschließlich Acknowledge-Pakete, also TCP-Bestätigungen ohne Nutzlast, sowie Application Data, die verschlüsselte Daten enthalten, beobachten. Eine detaillierte Untersuchung einzelner Streams mittels *Rechtsklick auf die Daten* > *Folgen* > *TCP-Stream* bestätigte, dass die übertragenen Inhalte verschlüsselt sind. Darüber hinaus ist im Dropdown-Menü Transport Layer Security ersichtlich, dass die verwendete TLS-Version 1.2 entspricht.

Durch den Filter `hci_h4` werden sämtliche Datenpakete erfasst, die über das HCI-Protokoll übertragen werden. Im Rahmen des Low Energy (LE) Meta-Events werden alle Ereignisse, die mit Bluetooth Low Energy in Verbindung stehen, gekapselt. Auf diese Weise übermittelt der Controller dem Host spezifische Informationen über BLE-bezogene Aktivitäten. Dabei wird deutlich, dass das untergeordnete Ereignis die Bezeichnung LE Extended Advertising Report trägt.

Die in diesem Kontext verwendeten Event-Typen des Schlosses sind Scannable, Scan Response und Legacy. Dies impliziert zum einen, dass das Gerät die Pairing-Version Legacy verwendet. Das genutzte Association Model – Just Works, Passkey Entry oder OOB – ist darüber jedoch nicht herauszufinden. Zum anderen deuten die Event-Typen daraufhin, dass das Schloss das klassische Legacy Advertising Format nutzt, welches eine maximale Datenlänge von 31 Byte aufweist. Zudem erlaubt das Gerät den Empfang von Scanning Requests durch Scanner – beispielsweise durch Smartphones – und reagiert darauf mit dem Versenden eines Scan Response-Pakets.

Der Peer Address Type gibt Aufschluss über die Art der vom werbenden Gerät genutzten Bluetooth-Adresse. Unter BR_ADDR ist die spezifische Bluetooth-Adresse (44:2c:xx:xx:xx:xx) erkennbar, die dem Schloss zugeordnet ist. In diesem Fall handelt es sich um eine Random Device Address, was bedeutet, dass verschiedene Unterarten dieser Adresskategorie existieren. Die binäre Schreibweise des Most Significant Bits (44) lautet 00101100. Bei Betrachtung der beiden höchstwertigen Bits, ergibt sich gemäß der Bit-Regel der Wert 00, was darauf hinweist, dass es sich um eine Non-Resolvable Private Address handelt. Eine solche Adresse wird zufällig generiert, regelmäßig geändert und kann nicht aufgelöst werden. [64]

Für die Datenübertragung kommt standardmäßig der LE 1M PHY-Modus zur Anwendung. Ein sekundärer Übertragungskanal wird dabei nicht genutzt. Die Angabe Transmit Power beschreibt die Sendeleistung des Gerätes. Der hier verzeichnete Wert von $127\,\mathrm{dBm}$ stellt einen Platzhalter dar und signalisiert, dass keine Messung vorliegt bzw. der Wert nicht übermittelt wurde. Das Feld Received Signal Strength Indicator gibt die Empfangsstärke am Controller wieder. Mit einem Wert von $-81\,\mathrm{dBm}$ ist diese als relativ schwach einzuordnen. Ein periodisches Advertising findet nicht statt.

Das Feld Direct Address Type verdeutlicht, dass das Gerät kein direktes Advertising betreibt. Folglich bleibt auch das zugehörige Feld der Bluetooth-Adresse leer.

Im Bereich der Werbedaten finden sich weitere Informationen: Hier wird zunächst die Länge des Advertising-Data-Elements in Byte angegeben. Die enthaltene 16-Bit-UUID 0xFEF3 weist darauf hin, dass es sich um eine herstellerspezifische UUID handelt, die von der Bluetooth SIG für Google LLC reserviert wurde [65]. Sie dient proprietären Diensten von Google. Der hersteller- bzw. dienstspezifische Payload lautet: 4a xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx 62. Da für diese Daten keine öffentlichen Spezifikationen verfügbar sind, ist eine detaillierte Auswertung nicht möglich.

Nach der Anwendung des Filters `btatt` konnten sämtliche ATT-Operationen identifiziert werden, die für die Analyse der BLE-Kommunikation von Relevanz sind. Die in diesem Zusammenhang erfassten Operationen werden im Folgenden systematisch dargestellt, wobei in der Tabelle 4.1 zusätzlich die Richtung der Kommunikation zwischen Client und Server sowie der jeweilige Opcode der Operationen angegeben ist. Es ist zu bemerken, dass der Opcode mit dem aus der Bluetooth Core Specification 4.2 übereinstimmt. Die an der Kommunikation teilnehmenden Geräte sind das Google Pixel 7a und das Yale Linus® Smart Lock mit jeweils den Bluetooth-Adressen 94:45:xx:xx:xx:xx beziehungsweise 78:9c:xx:xx:xx:xx.

**Tabelle 4.1:** ATT-Operationen und deren Bedeutung, basierend auf: [55]

| ATT-Operation | Beschreibung | Opcode | Quelle | Ziel |
|---|---|---|---|---|
| Read By Group Type Request | Fragt nach Primary Services (0x2800) in einem Handle-Bereich | 0x10 | Smartphone | Schloss |
| Read By Group Type Response | Antwortet mit allen Services, inklusive Start- und End-Handle sowie UUID | 0x11 | Schloss | Smartphone |
| Read By Type Request | Fragt nach Characteristics (0x2803) in einem Handle-Bereich | 0x08 | Smartphone | Schloss |
| Read By Type Response | Antwortet mit allen Characteristics, inklusive Handle, Service UUID, Value Handle, Properties und UUID | 0x09 | Schloss | Smartphone |
| Find Information Request | Fragt nach Attributen in einem Handle-Bereich | 0x04 | Smartphone | Schloss |
| Find Information Response | Antwortet mit Handles und zugehöriger UUID | 0x05 | Schloss | Smartphone |
| Read Request | Fragt nach dem Wert eines einzelnen Attributs | 0x0A | Smartphone | Schloss |
| Read Response | Antwortet mit dem Wert eines einzelnen Attributs | 0x0B | Schloss | Smartphone |
| Write Request | Schreibt einen Wert in ein Attribut und erwartet eine Bestätigung | 0x12 | Smartphone | Schloss |
| Write Response | Bestätigt die erfolgreiche Durchführung des Write Requests | 0x13 | Schloss | Smartphone |
| Write Command | Schreibt ein Wert in ein Attribut und erwartet keine Bestätigung | 0x52 | Smartphone | Schloss |
| Handle Value Indication | Informiert über die Änderung eines Attributwerts | 0x1D | Schloss | Smartphone |
| Handle Value Confirmation | Bestätigt den Empfang der Indication | 0x1E | Smartphone | Schloss |

Wie bereits in Kapitel 3.3.1 erläutert, war es mithilfe der Filter `btatt.opcode == 0x11` und `btatt.opcode == 0x09` möglich, die Anzahl der vorhandenen Services sowie der zugehörigen Characteristics zu ermitteln. Insgesamt konnten acht Services und 36 Characteristics identifiziert werden. Der GAP-Service umfasst die beiden Characteristics „Device Name“ und „Appearance“. Der GATT-Service weist hingegen keine Characteristics auf. Der Service „Device Information“ beinhaltet die Characteristics „Manufacturer Name String“, „Model Number String“, „Serial Number String“ sowie „Firmware Revision String“. Der Service „August Home Inc.“ verfügt über sechs Characteristics, die als „Unknown“ bezeichnet sind. Zusätzlich wurden fünf weitere, nicht näher spezifizierbare Services festgestellt, die jeweils acht, drei, fünf, fünf und drei unbekannte Characteristics enthalten.

Die Abbildung 4.1 zeigt eine schematische Darstellung aller Services mit deren zugehörigen Characteristics.

```
GAP (0x0001 - 0x0005) UUID: 0x1800
├── Device Name (0x0002) UUID: 0x2a00, Properties: 0x02
└── Appearance (0x0004) UUID: 0x2a01, Properties: 0x02

GATT (0x0006 - 0x0006) UUID: 0x1801

Device Information (0x0007 - 0x0007) UUID: 0x180a
├── Manufacturer Name String (0x0009) UUID: 0x2a29, Properties: 0x02
├── Model Number String (0x000b) UUID: 0x2a24, Properties: 0x02
├── Serial Number String (0x000c) UUID: 0x2a25, Properties: 0x02
└── Firmware Revision String (0x000f) UUID: 2x2a26, Properties: 0x02
August Home Inc. (0x0010 - 0x0020) UUID: 0xfe24
├── Unknown (0x0012) UUID: 6xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxd, Properties: 0x0c
├── Unknown (0x0014) UUID: 6xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxd, Properties: 0x20
├── Unknown (0x0017) UUID: 6xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxd, Properties: 0x0c
├── Unknown (0x0019) UUID: 6xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxd, Properties: 0x20
├── Unknown (0x001c) UUID: 6xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxd, Properties: 0x14
└── Unknown (0x001f) UUID: 6xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxd, Properties: 0x14
Unknown (0x0021 - 0x0039) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0
├── Unknown (0x0023) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
├── Unknown (0x0026) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
├── Unknown (0x0029) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
├── Unknown (0x002c) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
├── Unknown (0x002f) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
├── Unknown (0x0032) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
├── Unknown (0x0035) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x2a
└── Unknown (0x0039) UUID: dxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx6, Properties: 0x02
Unknown (0x003a - 0x0042) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0
├── Unknown (0x003c) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
├── Unknown (0x003f) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
└── Unknown (0x0042) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x02
Unknown (0x0043 - 0x0051) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0
├── Unknown (0x0045) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
├── Unknown (0x0048) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
├── Unknown (0x004b) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
├── Unknown (0x004e) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
└── Unknown (0x0051) UUID: dxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx6, Properties: 0x02
Unknown (0x0052 - 0x0062) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0
├── Unknown (0x0054) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
├── Unknown (0x0057) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x2a
├── Unknown (0x005b) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x2a
├── Unknown (0x005f) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
└── Unknown (0x0062) UUID: dxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx6, Properties: 0x02
Unknown (0x0063 - 0x006b) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0
├── Unknown (0x0065) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
├── Unknown (0x0068) UUID: 9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0, Properties: 0x0a
└── Unknown (0x006b) UUID: dxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx6, Properties: 0x02
```

**Abbildung 4.1:** Zuordnung der Charakteristiken zu ihren Services

Darüber hinaus werden weitere relevante Informationen veranschaulicht, wie etwa der Name, das Start- und End-Handle sowie die UUID der Services. Standardisierte UUIDs, die von der Bluetooth SIG definiert wurden, besitzen eine Länge von 16 Bit, während herstellerspezifische, proprietäre UUIDs eine Länge von 128 Bit aufweisen [55]. Für die Characteristics sind ergänzend der Name, das Characteristic Value Handle, die UUID sowie die Properties dargestellt. Die Bedeutungen der Properties, also welche Funktionen einer Characteristic zugeordnet sind, werden in der nachfolgenden Tabelle 4.2 erläutert. Die Zuordnung einer Characteristic zu dem jeweiligen Service erfolgte über das Characteristic Value Handle. Dieses liegt innerhalb des Bereichs zwischen dem Start- und End-Handle eines Services. Zudem sind Characteristics mit Deskriptoren, dessen Werte gesetzt sind, blau hervorgehoben. Durch den Abgleich dieser Werte war es möglich, eine Zuordnung vorzunehmen, die Übersicht zu erstellen sowie einen Abgleich mit dem Bluetooth ATT-Server-Attributprotokoll durchzuführen.

**Tabelle 4.2:** Die Werte der Characteristc Properties und deren Bedeutung für die einzelne Characteristic, basierend auf: [55]

| Characteristic Property | Bedeutung |
|---|---|
| 0x01 | Notify |
| 0x02 | Read |
| 0x14 | Notify, Write without Response |
| 0x0a | Write, Read |
| 0x0c | Write, Write without Response |
| 0x20 | Indicate |
| 0x2a | Indicate, Write, Read |

Durch die Anwendung des Filters `btatt` in Kombination mit dem regulären Ausdruck Read Response konnten die Werte der Characteristics eindeutig ermittelt werden. Dabei zeigte sich, dass die Seriennummer des Schlosses LxxxxxxxxT lautet und die Firmware-Version 3.2.2 beträgt.

Aus der Analyse aller aufgezeichneter BLE-Kommunikationsdaten zwischen dem Smartphone und dem Schloss, die verschiedene ausgeführte Aktionen – einschließlich Steuerungsbefehle sowie Gästeverwaltungsfunktionen – betreffen, lassen sich bestimmte Muster und Strukturen erkennen. Nach Abschluss des Advertisings und einem erfolgreichen Verbindungsaufbau übermittelte das Schloss dem Smartphone zunächst alle verfügbaren Services, Characteristics sowie weitere Attribute. Diese initiale Übertragung bildet die Grundlage für die spätere Ausführung spezifischer Aktionen.

Für die eigentliche Durchführung der Aktionen sind insbesondere die Write Requests mit dem Opcode 0x12 sowie die Write Commands mit dem Opcode 0x52 von Relevanz. Dies rechtfertigt die Anwendung des Filters `btatt.opcode == 0x12 || btatt.opcode == 0x52` bei der Analyse des Traffics. Im weiteren Verlauf ist zu beobachten, dass zunächst die Werte der CCCDs gesetzt werden. So erhalten die CCCDs an den Handles 0x0015 und 0x001a den Wert 0x0002, was dem Status „Indicate“ entspricht. Dies bedeutet, dass die zugehörige Characteristic nach Ausführung einer Operation eine Indication und die zugehörige Confirmation erfordert. Der CCCD an Handle 0x0002 wird mit dem Wert 0x0001 konfiguriert, was dem Status „Notification“ entspricht und lediglich eine einfache Benachrichtigung ohne Rückmeldung erzeugt.

Im Anschluss daran lassen sich spezifische Schreibvorgänge beobachten: Jeweils zwei Write Requests werden an Handle 0x0017 gesendet, gefolgt von einer Write Response auf demselben Handle. Darauf erfolgen jeweils eine Handle Value Indication sowie eine Handle Value Confirmation an Handle 0x0019. Dieses Verhalten deutet darauf hin, dass Handle 0x0017 primär der Authentifizierung dient, während Handle 0x0019 die Statusbestätigung der Authentifizierungsvorgänge übernimmt.

Daraufhin werden mindestens vier Write Requests an Handle 0x0012 gesendet, begleitet von einer gleich großen Anzahl an Write Responses. Auch hierbei folgen nach jedem Write Response eine Handle Value Indication sowie eine Handle Value Confirmation an Handle 0x0014. Dieses Muster legt nahe, dass Handle 0x0012 die Steuerung der eigentlichen Aktionen übernimmt, während Handle 0x0014 die Statusbestätigung der ausgeführten Aktionen bereitstellt. Die Write-Operationen legen somit die auszuführenden Aktionen fest, während Indications und Confirmations die Verarbeitung dieser Befehle durch das Schloss widerspiegeln (siehe Tabelle 4.1).

Die Analyse zeigt außerdem, dass die Ausführung der Aktionen in mehreren Paketen erfolgt, da die maximale Datenlänge eines BLE-Pakets auf 31 Byte begrenzt ist. Die Values, die zur Steuerung der Aktionen verwendet werden, sind nicht fest vorgegeben — beispielsweise 0x00 für „geschlossen“ oder 0x01 für „geöffnet“ — sondern variieren kontinuierlich, sodass kein wiederkehrendes Muster erkennbar ist.

## 4.2 Auswertung der Cloud-Daten

In der ersten Zeile der heruntergeladenen Comma Separated Values (CSV)-Datei sind sämtliche allgemeinen Bezeichnungen der nachfolgenden Variablen enthalten. Diese sind kommasepariert voneinander aufgelistet und strukturieren die in den folgenden Zeilen enthaltenen Daten. Die relevanten Variablen sind folgende:

- Time
- Action
- User
- Phone Number
- Email Address
- Device Identification Number
- Location

Im Feld „Time“ wird der Zeitpunkt des jeweiligen Ereignisses auf die Sekunde genau dokumentiert. Die Angabe erfolgt in Universal Time Coordinated (UTC) +0. Da die Datenerhebung am 28.07.2025 durchgeführt wurde, also während der europäischen Sommerzeit, ist für eine korrekte zeitliche Einordnung der Ereignisse eine Umrechnung erforderlich. Für Deutschland ergibt sich somit ein Zeitversatz von Universal Time Coordinated (UTC) +2, sodass zwei Stunden addiert werden müssen. Das Feld „Action“ beschreibt, welche konkrete Handlung ausgeführt wurde, beispielsweise das manuelle oder automatische Ver- und Entriegeln des Schlosses oder das Zuweisen von Gastzugängen. Änderungen von Einstellungen (bspw. das

Aktivieren der Auto Lock Funktion und dazugehörigen Parametern oder von Smart Alerts) werden jedoch nicht erfasst, ebenso wenig wird die genaue Art eines vergebenen Gastzugangs (z. B. temporär, wiederkehrend, immer oder Eigentümer) aufgeführt. Im Feld „User“ wird der Name der Person angegeben, die die jeweilige Handlung vorgenommen hat, wodurch sich Aktionen eindeutig auf einzelne Nutzer zurückführen lassen. Das Feld „Phone Number“ dokumentiert die Telefonnummer der handelnden Person. Diese bleibt jedoch leer, wenn die Aktion ohne persönliche Interaktion durchgeführt wurde, etwa beim automatischen Verriegeln oder beim manuellen Öffnen des Schlosses über den Drehknauf. Die Angabe „Email Address“ erfolgt ausschließlich für den Eigentümer selbst und stimmt mit der tatsächlich verwendeten Adresse überein. Auch in diesem Fall erfolgt ohne persönliche Interaktion keine Angabe der Variable. Gastnutzer erhalten in diesem Feld keinen Eintrag. Die „Device Identification Number“ stellt die eindeutige Kennung des Geräts dar. Das Feld „Location“ gibt die Zeitzone in der Form Kontinent/Stadt an und folgt dabei der Internet Assigned Numbers Authority (IANA)-Bezeichnung für Zeitzonen [66]. Dadurch lässt sich der Standort und der zeitliche Kontext eines Ereignisses eindeutig einordnen.

Allgemein lässt sich feststellen, dass beim manuellen Öffnen über den Drehknauf weder Telefonnummer noch E-Mail Adresse erfasst werden, was nachvollziehbar ist, da es sich um eine direkte, nicht automatisierte Nutzerinteraktion handelt. Ebenso fehlen diese Angaben bei automatischen Vorgängen wie der Auto Lock Funktion. Änderungen an Einstellungen, beispielsweise die De-/Aktivierung von Auto Lock bzw. Auto Unlock, sind im Protokoll nicht dokumentiert. Wenn mehrere Schlösser installiert sind, kann zudem die jeweilige Tür über den vergebenen Namen eindeutig identifiziert werden. Die Seriennummer für das in diesem Fall verwendete Schloss lautet LxxxxxxxxT und ist in allen Einträgen nachzulesen.

Die Analyse des Ereignisprotokolls zeigt, dass das reine Einsehen und Aktualisieren der Einträge durch den Hauptnutzer keine zusätzlichen Protokolleinträge erzeugt. Bei der Aktion 21 ist eine Inkonsistenz zwischen dem händisch erfassten Protokoll und den in der Cloud gespeicherten Daten festzustellen. Während in dem selbst erstellten Protokoll die Entriegelung durch den Gastnutzer als Vorgang über BLE dokumentiert wurde, wird derselbe Vorgang in den Cloud-Daten als Fernöffnung ausgewiesen. Die Aktionen 24, 25 und 26 sind im Protokoll nicht enthalten. In diesem Zeitraum war es dem Gastnutzer dennoch möglich, das Schloss trotz des zuvor erfolgten Entzugs seiner Berechtigungen zu betätigen. Die Aktion 29, das Entriegeln durch den Hauptnutzer, wurde in den Daten als Fernöffnung erfasst, obwohl sich das Gerät in unmittelbarer Nähe befand und eine Kommunikation über BLE möglich gewesen wäre. Die Aktion 31, bei der das Schloss zuvor aus der Ferne entriegelt wurde und die Auto Lock-Funktion auf zehn Sekunden eingestellt war, ist im Ereignisprotokoll nicht verzeichnet. Die Aktion 34 ist in den Cloud-Daten nicht als einzelner Vorgang gespeichert, sondern in vier separate Ereignisse aufgeteilt (zweimal manuelles Entriegeln sowie zweimal manuelles Verriegeln).

Hinsichtlich der Nutzerinformationen ist festzustellen, dass die E-Mail-Adresse des Eigentümers gespeichert wird und mit der tatsächlich verwendeten Adresse übereinstimmt. Für Gastnutzer werden hingegen keine E-Mail-Adressen gespeichert. Telefonnummern von Haupt- und Gastnutzern werden hingegen erfasst. Die im Protokoll dokumentierten Zeiten weichen geringfügig von denen in der Cloud ab.

Die vorliegenden Daten erwiesen sich insgesamt als wertvolle Grundlage für eine weiterführende forensische Analyse. Sie ermöglichen es, zeitliche Abfolgen von Aktionen präzise nachzuvollziehen, Nutzerhandlungen voneinander zu unterscheiden und Zugriffsrechte zu rekonstruieren. Darüber hinaus dienen sie dem Abgleich zwischen dem eigenen Protokoll, der App-internen Ereignisaufzeichnung und der exportierten CSV-Datei, wodurch die Nachvollziehbarkeit und Validität der Informationen erheblich erhöht wird. Diese CSV-Datei ist somit auch als Ereignisprotokoll zu verstehen und erlaubt daher einen direkten Vergleich mit den zwei weiteren Protokollen. Eine Übersicht dessen ist in Tabelle B.1 bis B.4 zu sehen.

## 4.3 Analyse der App-Daten

Zur Auswertung der App-spezifischen Daten, die im Dateisystem des Google Pixel 7a abgelegt sind, ist zunächst festzuhalten, dass nicht alle extrahierten Dateien relevante Informationen enthalten und auf einige aufgrund von Verschlüsselung nicht zugegriffen werden konnte. Im Folgenden werden die gewonnenen Erkenntnisse dargestellt. Doppelt vorkommende Einträge werden nur einmal genannt.

Aus dem Google Pixel 7a konnten die nachstehenden Daten extrahiert werden. Im Verzeichnis *shared_prefs* befinden sich Schlüssel-Wert-Paare in XML-Dateien, die Rückschlüsse auf Konfigurationen zulassen. In der Datei *com.google.android.gms.measurement.prefs.xml* werden sowohl Einstellungen für Mess- und Analysedienste als auch Präferenzen des Benutzers gespeichert, die sich auf die Google-Play-Dienste beziehen. Enthalten sind unter anderem: die unter Android verwendete Firebase-App-ID (1:824469303174:android:33bc5f167fde92a431c607), das Datum des erstmaligen App-Starts sowie des letzten Übergangs in den Hintergrund (jeweils in Mitteleuropäischer Normalzeit vermerkt), die Betriebssystemversion des verwendeten Smartphones sowie die aktuelle Session-ID. Diese Session-ID dient der Analyse durch Firebase und besitzt keine Steuerungsbefugnisse [67].

Darüber hinaus werden spezifische Konfigurations- und Statusinformationen festgehalten wie:

- ob die App zuletzt im Hintergrund ausgeführt wurde (boolean: true),
- ob die App nach der Installation mindestens einmal geöffnet wurde, (boolean: true)
- ob personalisierte Werbung deaktiviert wurde (string: true),
- ob beim nächsten App-Start eine neue Session erzwungen werden soll (boolean: true) und
- die Zustimmung zu den Cookie-Einstellungen (string: G100 → Cookies für Tracking-Zwecke abgelehnt).

Die Datei *WebViewChromiumPrefs.xml* enthält die zuletzt verwendete Versionsnummer der WebView-Komponente. In der Datei *com.august.luna.start.SHARED_PREFERENCES.xml* sind unter den Zeichenketten *START_FAVORITE_HOUSE* und *START_FAVORITE_LOCK* jeweils eine UUID für das bevorzugte Haus sowie eine alphanumerische Zeichenfolge für das priorisierte Schloss abgelegt, die diesen eindeutig zuzuordnen sind. Die UUID für das Haus lautet 7xxxxxxx-xxxx-

xxxx-xxxx-xxxxxxxxxx0 und die Zeichenkette für das Schloss BxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxC. Bei der Nutzung mehrerer Schlösser an unterschiedlichen Häusern ermöglicht dies eine eindeutige Zuordnung. Darüber hinaus liefert diese Speicherung zusätzlichen Kontext.

In der zuvor genannten Datei sowie in der Datei *com.august.luna.xml* sind die Klarnamen *august* und *luna* enthalten. Dies kann darauf hindeuten, dass an dieser Stelle weitere schlossbezogene Daten persistiert werden. In letzterer Datei finden sich darüber hinaus folgende Informationen: eine während der Installation der App generierte ID, eine fünfstellige Session-ID, eine sechsstellige Session-ID sowie eine Flow-ID, die der Nachverfolgung von Schlossereignissen dienen. Der funktionale Unterschied zwischen diesen IDs lässt sich nicht eindeutig bestimmen, da weder öffentliche Dokumentationen seitens des Herstellers noch Vergleichsdaten anderer Schlosssysteme vorliegen. Zusätzlich beinhaltet die XML-Datei den Unix-Zeitstempel des Installationsdatums der App sowie den Zeitpunkt der erstmalig erfolgreichen Nutzung des Schlosses.

In der Publikation Security Analysis of Internet-of-Things: A Case Study of August Smart Lock wird dargelegt, dass die App August Home den für den Handshake benötigten Schlüssel, insbesondere auf gerooteten Endgeräten, im Klartext innerhalb des Verzeichnisses */data/data/com.august.app/shared_prefs/* in der Datei *PeripheralInfoCache.xml* ablegt [68]. Ein direkter Vergleich mit den hier vorliegenden Ergebnissen ist jedoch nicht möglich, da in diesem Fall unter *__androidx_security_crypto_encrypted_prefs_key_keyset__* der verschlüsselte MasterKey gespeichert ist. Dieser dient dem Schutz des Value-Keys, welcher wiederum unter *__androidx_security_crypto_encrypted_prefs_value_keyset__* abgelegt wird. Der Value-Key verschlüsselt schließlich die in den Preferences enthaltenen Schlüssel-Wert-Paare.

Im Verzeichnis `\files\logs` befinden sich die Log-Dateien, die nach Tagen in Textdokumente unterteilt sind. Diese Dateien sind nach dem Format *log.jahr-monat-tag.0.txt* benannt. Es ist festzustellen, dass neben der Datei vom 28.07.2025 auch die Datei *log.txt* dem Tag der Datenerhebung zuzuordnen ist und ebenfalls Log-Einträge enthält. Beide Dateien gehen nahtlos ineinander über, ohne dass eine Lücke in der Aufzeichnung erkennbar ist. Der Aufbau der Dateien gestaltet sich wie folgt: Ein Log-Eintrag besteht aus einem Zeitstempel, der das Datum und die Uhrzeit gemäß der geltenden Zeitzone (hier: Mitteleuropäische Sommerzeit) bis auf die Tausendstelsekunde genau enthält. Alle Einträge sind chronologisch geordnet. Im Anschluss an den Zeitstempel folgt das Log-Level (Info, Debug, Warn oder Error), abhängig von der Art des jeweiligen Ereignisses. Daraufhin wird das Ereignis inhaltlich beschrieben. Darüber hinaus konnte festgestellt werden, dass die Kommunikation zwischen Server und Client über das Message Queuing Telemetry Transport-Protokoll erfolgt. Der zugehörige Publish-Subscribe-Kanal ist in der Datenbank *devices* hinterlegt und trägt die Bezeichnung 2295c5da-8427-4e4f-b6bb-f44249eaa6e0.

Bei der Analyse der beiden Dateien fällt auf, dass mittels der Suchfunktion keine der zuvor ermittelten Session-IDs aufgefunden werden konnte. Mithilfe der Suchfunktion ließ sich jedoch feststellen, dass in den Log-Dateien das Auslösen eines Smart Alerts dokumentiert wird. So konnte ermittelt werden, dass auf die Aktion 20 (Aktivierung des Smart Alerts „Gerät wird von einem bestimmten Benutzer ent- oder verriegelt“) ein Eintrag erzeugt wird, dem eine eindeutige Benachrichtigungs-ID zugewiesen ist. Der Eintrag enthält zudem die ID und den Namen des Schlosses, die ID der ausführenden Person (Gastnutzer) sowie deren Namen

```
2025-07-28 11:07:14.686 GMT+02:00 DEBUG T: Firebase-Messaging-Intent-Handle
GcmMessageHandlerV2 - received a v2 push notification!
{messageID=68873da431550ca39a60902a, status=lock_operation,
extra={"userID":"e4d3f505-b9ab-4962-93a1-4d9d2c41feb9",
"deviceType":"lock","type":6, "deviceID":"BEA3235E63670A907A393E26F1E1181C",
"deviceName":"VORDERTÜR", "action":"unlock"}, title=Yale Home -
Nutzeraktivität, version=2, message=Google User hat VORDERTÜR in Reallabor
entriegelt., houseName=Reallabor, houseID=79b851ae-fe48-4135-b9e1-84cfc41d6590}
```

**Abbildung 4.2:** Log-Eintrag beim Auslösen des Smart Alerts

(Google User). Darüber hinaus werden die ausgeführte Aktion und die an den Hauptnutzer versendete Benachrichtigung dokumentiert. Ebenso sind die ID des Hauses sowie dessen Bezeichnung im Log vermerkt. Der Log-Eintrag ist in der Abbildung 4.2 zu sehen.

In dem Eintrag, der die Antworten auf ein Ereignis des Schlosses zusammenführt, wird zwischen den Ereignissen „Bluetooth Connection“ und „UI Bluetooth Event“ unterschieden. Weiterhin werden darin die Version des Software Development Kits, der Herstellername, die exakte Gerätebezeichnung des Smartphones des Hauptnutzers sowie die Yale Home App-Version erfasst.

Im Zusammenhang mit dem über den BLE Lock Manager gesteuerten BLE-Verbindungsstatus werden zusätzlich die Seriennummer, die Firmware-Version sowie die teilweise anonymisierte Bluetooth-Adresse des Schlosses (wobei die letzten zwei Oktette sichtbar bleiben) gespeichert. Der Verbindungsstatus kann dabei folgende Zustände annehmen: disconnecting, disconnected, connecting, connected oder ready. Darüber hinaus wird die verstrichene Zeit in Millisekunden gemessen. Ferner konnte ein Handshake-Schlüssel identifiziert werden. Zudem wird die Seriennummer der Bridge erfasst, wenn ein Update dieser Komponente durchgeführt wurde.

Im Verzeichnis *databases* befinden sich Datenbanken, die Informationen zu Nutzerdaten und -einstellungen sowie zu den beteiligten Geräten enthalten. In der Datenbank *ModelDatabase.db* sind die Tabellen *BridgeData*, *HouseData* und *LockData* abgelegt. Diese speichern zum einen die ID, die Firmware-Version, die Seriennummer sowie das Modell der Bridge und zum anderen die ID und den Namen des Hauses. Darüber hinaus werden die ID, die Seriennummer, die Firmware-Version, die Stock Keeping Unit (SKU), der Name sowie die Bluetooth-Adresse des Schlosses erfasst. Tabelle 4.3 liefert eine detaillierte Übersicht über die konkret gespeicherten Werte.

In derselben Datenbank befindet sich die Tabelle *UserData*, in der Informationen über den Hauptnutzer sowie alle Gastnutzer gespeichert sind. Zu diesen Informationen zählen die ID, der Vor- und Nachname sowie eine r1-Nummer. Zusätzlich werden für den Hauptnutzer die Region sowie Angaben zum Vorhandensein einer verifizierten E-Mail-Adresse und eines Passworts erfasst. Die doppelte Erfassung eines Gastnutzers lässt darauf schließen, dass während des Versendens der Zugangsberechtigung durch den Hauptnutzer zunächst der erste Eintrag erzeugt wurde und durch die Anmeldung des Gastnutzers in der Yale Home App über dessen bestehendes Google-Konto ein zweiter Eintrag hinzugefügt wurde. Daraus ergibt sich die Annahme, dass die Spalte „r1“ Relationen zwischen den Nutzereinträgen abbildet, da die entsprechende Nummer für die Nutzer der zweiten und dritten Einträge identisch ist.

Darüber hinaus existiert eine Spalte „lockRules“, welche die Zuordnung der jeweiligen Schloss-ID zu einer spezifischen Zugangsberechtigungsregel abbildet. In Tabelle 4.4 sind sämtliche Informationen übersichtlich dargestellt.

**Tabelle 4.3:** Informationen zu der Bridge, dem Haus und dem Schloss aus den Tabellen *BridgeData*, *HouseData* und *LockData* der Datenbank *ModelDatabase.db*

| **Kategorie** | **BridgeData** |
|---|---|
| ID | 6xxxxxxxxxxxxxxxxxxxxx7 |
| Firmware-Version | 2.4.0 |
| Seriennummer | CxxxxxxxxR |
| Modell | august-connect |

| **Kategorie** | **HouseData** |
|---|---|
| ID | 7xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx0 |
| Name | Reallabor |

| **Kategorie** | **LockData** |
|---|---|
| ID | BxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxC |
| Firmware-Version | 3.2.2 |
| Seriennummer | LxxxxxxxxT |
| Name | VORDERTÜR |
| SKU | ASL6_05/101510/SI |
| Bluetooth-Adresse | 78:9C:xx:xx:xx:xx |

**Tabelle 4.4:** Informationen zu den Nutzern der Yale Home App aus der Tabelle *UserData* der Datenbank *ModelDatabase.db*

| **Kategorie** | **UserData 1** |
|---|---|
| ID | 6d4233ea-678f-4288-b83f-c5ee79137d17 |
| Vorname | Real |
| Nachname | Labor |
| r1 | iVE0pDHsJ/FGs2n/]NNgXEQ/d8VCw95c+ nxIprQyaSLvm6su8k3cH3c2J |
| Region | Deutschland |
| E-Mail | vorhanden |
| Passwort | vorhanden |

| **Kategorie** | **UserData 2** |
|---|---|
| ID | phone:+49xxxxxxxxx0 |
| Name | Emelie Hanske |
| r1 | zGlwGiZWSbv1T6yL]F0JDIXlWoB1AUnVp i5eq9QkOHCicnIXFO7GwaVk= |

| **Kategorie** | **UserData 3** |
|---|---|
| ID | e4d3f505-b9ab-4962-93a1-4d9d2c41feb9 |
| Vorname | Google |
| Nachname | User |
| r1 | zGlwGiZWSbv1T6yL]F0JDIXlWoB1AUnVp i5eq9QkOHCicnIXFO7GwaVk= |

In der Tabelle *RuleData* werden Einträge für die verschiedenen erstellten Zugangsberechtigungsregeln angelegt. Jede dieser Regeln erhält eine eindeutige ID. Darüber hinaus sind das Start- und Enddatum der Regel dokumentiert, wobei die zum Zeitpunkt der Datenerhebung gültige Mitteleuropäische Sommerzeit berücksichtigt wurde. In der Spalte „recurrence“ sind die Einstellungen der jeweiligen Regel hinterlegt, das heißt, ob sie wiederkehrend, temporär, dauerhaft oder als Eigentümerregel definiert ist. Für die während der Datenerhebung getesteten Zugangsberechtigungen wurde ausschließlich ein Eintrag erstellt, und zwar für die wiederkehrende Zugangsberechtigung. Dies ist an der angegebenen Frequenz erkennbar, welche mit wöchentlich für den Tag Montag gekennzeichnet ist.

In der Tabelle *UserLocation* der Datenbank *LocationDatabase.db* sind Informationen im Zusammenhang mit der Verwendung der Funktionen Auto Lock und Auto Unlock gespeichert. Die Einträge enthalten jewels einen Zeitstempel im Unix-Format, die ID des zugehörigen Schlosses sowie die durchgeführte Aktion. Die Informationen zu den ausgeführten Aktionen werden als Dictionary im Format `„a“:„getätigte Aktion“,„b“:„Name des Schlosses/der Person“` gespeichert. Aufgrund zuvor gespeicherter Einträge ist bekannt, dass sich die unter „a“ aufgeführten Aktionen auf folgende belaufen können:

- zu Hause verlassen
- zu Hause angekommen
- Zone verlassen
- in der Zone
- automatisch geöffnet
- Funktion Auto Unlock aktiviert
- Funktion Auto Unlock deaktiviert

Unter der Variable „b“ wird entweder die ausführende Person „Reallabor“ oder der Name des betreffenden Schlosses „VORDERTÜR“ gespeichert. Für den Zeitraum der Datenerhebung wurde ein Eintrag identifiziert. Für das Yale Linus® Smart Lock mit der ID BEA3235E63670A907 A393E26F1E1181C wurde am Montag, den 28.07.2025 11:19:43 UTC +2 die Funktion Auto Unlock aktiviert. Diese Daten stimmen mit den dokumentierten Informationen im selbst erstellten Protokoll überein.

In der Tabelle *DeviceNetwork* der *LocationDatabase*-Datenabnk ist unter dem Spaltenname „Service Set Identifier“ der Name des verwendeten WLAN-Netzwerkes zu finden. Dieser stimmt mit dem verwendeten Netzwerk überein. Die Tabelle *locks* der Datenbank *devices* enthält Angaben zur Zeitzonenzuordnung „Europe/Berlin“ sowie zum Namen des jeweiligen Hauses (hier: Reallabor).

Die *last-exit-info*-Datei im Verzeichnis `\app_webview` liefert Informationen zum Zustand der App zum Zeitpunkt des Schließvorgangs. Dokumentiert werden hierbei die Identifikationsnummer des Beendigungsprozesses der Anwendung (exitInfoPid: 17122), der entsprechende Zeitstempel (timestampAtLastRecordingInMillis: 1730460662248) sowie der App-Status in Form eines numerischen Wertes (appState:3).

Die Datei *token_digest.preferences_pb* im Verzeichnis `\files\datastore` enthält den Hashwert 3xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 eines Tokens. Im selben Verzeichnis befindet sich die Datei *firebase_session_Y29tLmFzc2FhYmxveS55YWxl_settings.preferences_pb* in welcher folgende Session-ID gespeichert ist: fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx6.

Im Verzeichnis `\files\.com.google.firebase.crashlytics.files.v2_com.assaabloy.yale\open-sessions\6887458B01F20001359554153FD124BD` befindet sich eine *report*-Datei, welche sowohl die verwendete Software Development Kit-Version als auch die Build-Version dokumentiert. Diese entsprechen den Versionen 19.0.2 und 250714471. In diesem Zusammenhang wurde zusätzlich folgende Kennung ermittelt: Nxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx=.

Im Unterverzeichnis `\native` liegen vier JSON-Dateien vor: *app.json*, *device.json*, *os.json* und *session.json*. Diese Dateien enthalten Informationen zur Anwendung, zum Gerät, zum Betriebssystem sowie zur Sitzung. In *app.json* sind der Versionsname (2025.8.0), die Kennung (com.assaabloy.yale) und die Installation-UUID (6xxxxxxxxxxxxxxxxxxxxxxxxxxxx7) der Yale Home App gespeichert. Die Datei *device.json* enthält Angaben zu den Hardwarekomponenten und Build-Details des verwendeten Endgeräts. Insbesondere werden die Anzahl der verfügbaren Prozessorkerne, die Prozessorarchitektur, der verfügbare Gerätespeicher sowie der Arbeitsspeicher aufgeführt. Die Build-Informationen weisen darauf hin, dass es sich bei dem analysierten Testgerät um ein Google Pixel 7a handelt. Die Datei *os.json*-Datei dokumentiert, dass die Build-Version der öffentlich freigegebenen Softwareversion Android 14 entspricht. In der Datei *session.json* sind sowohl der Startzeitpunkt der Sitzung als auch die zugehörige Sitzungs-ID vermerkt.

Im Verzeichnis `\priority-reports` befindet sich die Datei *686CE63802F2000157D154153FD124BD*, welche die Prozess-ID beinhaltet. Die Datei *com.crashlytics.settings.json*, die sich ebenfalls im Verzeichnis `\files\.com.google.firebase.crashlytics.files.v2_com.assaabloy.yale` befindet, liefert Informationen zur Qualität der Sitzung. Die Abtastrate beträgt ein Sample pro Sekunde. Eine Sitzung wird automatisch beendet, sofern die App für 1800 Sekunden (entspricht 30 Minuten) inaktiv bleibt.

## 4.4 Resultate des manuellen Vergleichs der Ereignisprotokolle

Die Protokollierung der ausgeführten Aktionen im App-Protokoll erfolgt unter dem jeweils aktuellen Datum, das als Überschrift dient. Unterhalb dieses Datums werden alle durchgeführten Aktionen in Listenform mit der zugehörigen Uhrzeit aufgeführt. Die Zeitangaben erfolgen, entsprechend der mitteleuropäischen Sommerzeit, im Format UTC+2. Im Folgenden werden die Ergebnisse der Auswertung des App-Protokolls detailliert dargestellt.

Aktion 11, das Entriegeln des Schlosses außerhalb des zulässigen Zeitraums, wird im Protokoll nicht erfasst; auch der Versuch der Ausführung ist nicht dokumentiert. Die Erstellung eines Smart Alerts (Aktion 20) findet ebenfalls keine Berücksichtigung. Aktion 21 ist im Protokoll als „Entriegeln aus der Ferne“ verzeichnet, obwohl die Aktion tatsächlich über BLE ausgeführt

wurde. Für das Entriegeln wurde eine Push-Benachrichtigung auf dem Telefon des Hauptnutzers gefunden. Das Verriegeln des Schlosses (Aktion 22) löste jedoch keine Benachrichtigung aus. Die Aktionen 24, 25 und 26 fehlen vollständig, obwohl 24 und 25 nachweislich ausgeführt wurden. Die Aktion 26 hingegen wies einen Autorisierungsfehler auf. Darüber hinaus ist die Aktivierung der Funktionen Auto Lock (Aktion 27) und Auto Unlock (Aktion 28) nicht protokolliert.

Für Aktion 29 werden zwei separate Einträge vorgenommen: Entriegeln und Verriegeln (über Auto Lock). Zwischen Aktion 32 und Aktion 33 ist zudem ein weiteres Entriegeln aus der Ferne erfasst. Aktion 35 wurde ebenfalls nicht dokumentiert. Bei der Funktion „Zugangsberechtigung an Gastnutzer erteilen“ wird die Art der Berechtigung (wiederkehrend, temporär, permanent, Eigentümer) nicht angegeben. Das Einsehen und Aktualisieren des Ereignisprotokolls erzeugt keinen Eintrag in diesem selbst. Die erfassten Zeitstempel weichen teilweise geringfügig, um bis zu eine Minute, von den tatsächlichen Ausführungszeiten ab.

Die Abbildungen in A.15 verdeutlichen die zuvor dargestellten Zusammenhänge. Sie dienen als Nachweis und bestätigen die im Text beschriebenen Ergebnisse.

# 5 Diskussion

In den folgenden Unterkapiteln werden zunächst die auf Grundlage der im Methodikteil beschriebenen und angewendeten Verfahren gewonnenen Ergebnisse zu den Netzwerk-, Cloud- und App-Daten sowie zum Ereignisprotokoll-Vergleich interpretiert. Anschließend werden die Grenzen der Arbeit dargestellt und die sich aus unterschiedlichen Umständen ergebenden Herausforderungen erläutert. Darauf folgt ein Ausblick auf Aspekte, die in zukünftigen Untersuchungen berücksichtigt werden sollten, um die forensische Analyse von IoT-Geräten - insbesondere Smart Locks - zielführend zu gestalten, weiterführende Erkenntnisse zu ermöglichen und einen Überblick über das Forschungsfeld der Smart Locks zu geben. Abschließend fasst das Fazit die zentralen Ergebnisse der Arbeit zusammen.

## 5.1 Interpretation der Ergebnisse

Die nachfolgenden Abschnitte widmen sich einer vertieften Analyse und Interpretation der zuvor präsentierten Ergebnisse zu den Netzwerk-, Cloud- und App-Daten sowie des manuellen Vergleichs der Ereignisprotokolle.

### 5.1.1 Netzwerk-Daten

Die Analyse der aufgezeichneten Netzwerkdaten weist einige Aspekte auf, die einer genaueren Einordnung bedürfen. Der Verbindungsaufbau zwischen der Cloud und der Bridge sowie der Aufbau der sicheren Verbindung ist im Protokoll nicht sichtbar, da dieser bereits stattgefunden hat, bevor die Aufzeichnung des Netzwerkverkehrs begann. Im weiteren Verlauf sind ausschließlich Application Data- und ACK-Pakete zu beobachten. Dies bedeutet, dass Nutzdaten an die Cloud oder die Bridge übertragen werden. Der Empfang dieser Pakete wird jeweils von der Gegenstelle durch ein ACK bestätigt. Das verwendete TLS-Protokoll weist darauf hin, dass die Inhalte verschlüsselt sind.

Während der Analyse fiel zudem auf, dass die IP-Adressen 3.33.245.2 und 15.197.218.233 der Amazon Technologies Inc. zugeordnet sind und ausschließlich mit der IP-Adresse der Bridge (10.0.1.138) kommunizieren. Aus der in Unterabschnitt 2.6.2 beschriebenen Systemarchitektur geht hervor, dass die Kommunikation vom Schloss über die Bridge und den Heimrouter an die Cloud weitergeleitet wird, von wo aus die Informationen schließlich das Smartphone erreichen. Daraus folgt, dass die beiden identifizierten IP-Adressen der Amazon Technologies Inc. die Cloud-Server darstellen, mit denen die Bridge über den Router hinweg Daten austauscht, um Informationen an das Smartphone weiterzugeben.

In der vorliegenden Aufzeichnung ist ausschließlich der Kommunikationsweg zwischen Bridge und Cloud enthalten. Dies erklärt, weshalb das Smartphone nicht sichtbar ist: Es wird, wie die Bridge, über die Cloud angebunden. Im Netzwerkprotokoll erscheinen daher nur die IP-Adressen der Bridge und der Cloud-Server, die jeweils die logischen Endpunkte der Kommunikation repräsentieren. Die Verbindungen bestehen dabei entweder von der Bridge zur ersten oder zweiten Cloud-Instanz oder in umgekehrter Richtung. Zwischen Bridge und Cloud

befindet sich jedoch noch der Heimrouter, der die Pakete entgegennimmt und entsprechend weiterleitet. In den Wireshark-Aufzeichnungen ist im Ethernet-II-Protokoll daher für alle externen IP-Adressen dieselbe MAC-Adresse zu sehen, da diese den nächsten physischen Hop im lokalen Netzwerk darstellt. Diese MAC-Adresse wird nicht über den Router hinaus übertragen. Dies legt zugrunde, dass die MAC-Adresse f4:90:ea:00:ff:f5 die LAN-MAC-Adresse des Routers darstellt, welche das Ziel für alle Geräte im lokalen Netzwerk bildet, die Datenpakete nach außen übertragen. Dies gilt somit auch, wenn die Bridge eine Nachricht an die Server der Amazon Technologies Inc. sendet. Die MAC-Adresse 70:61:be:c0:13:a7 hingegen entspricht der WAN-MAC-Adresse des Routers und fungiert als Ziel für alle Geräte im Internet, die Daten an ein Gerät innerhalb des lokalen Netzwerks übermitteln. Entsprechend trifft dies auch zu, wenn der Server eine Nachricht an die Bridge sendet. Eine direkte Kommunikation zwischen den IP-Adressen der Amazon Technologies Inc. findet nicht statt, da deren Aufgabe in der Speicherung, Lastverteilung und Redundanzsicherung der Daten im Hintergrund besteht.

Für den BLE-Traffic ergeben sich folgende Beobachtungen: Die Übermittlung der Services und Characteristics vom Schloss an die Smartphone-App ist erforderlich, damit das Schloss weiß, welche Aktionen ausgeführt werden dürfen (siehe Abschnitt 2.7). Einige dieser Services und Characteristics sind als `unknown` gekennzeichnet, was bedeutet, dass sie nicht von der Bluetooth SIG reserviert, sondern herstellerspezifisch sind. In diesem Fall von August Inc., da Yale Teil dieses Unternehmens ist [69]. Darüber hinaus zeigt sich, dass 16-Bit-UUIDs dem Bluetooth-SIG-Standard entsprechen, während 128-Bit-UUIDs herstellerspezifisch definiert werden. Auch diese sind hier August Inc. zuzuordnen.

### 5.1.2 Cloud-Daten

Die Analyse des Ereignisprotokolls zeigt, dass das bloße Einsehen und Aktualisieren von Einträgen durch den Hauptnutzer keine zusätzlichen Protokolleinträge erzeugt. Dies deutet darauf hin, dass derartige Lese- und Aktualisierungsvorgänge vom System nicht als sicherheitsrelevant eingestuft werden.

Das Nichtvorhandensein eines Eintrags zur Erstellung eines Smart Alerts im App-Protokoll erweist sich als problematisch, da dadurch zu einem späteren Zeitpunkt nicht mehr nachvollzogen werden kann, wann dieser eingerichtet wurde und ab welchem Zeitpunkt der entsprechende Sicherheitsmechanismus (z. B. eine Push-Benachrichtigung beim Ent- oder Verriegeln des Schlosses durch einen bestimmten Nutzer) wirksam war. Dies führt zu einer Lücke in der Nachvollziehbarkeit der ausgeführten Handlungen sowie in der eindeutigen Zuordnung dieser Aktion zu einer bestimmten Person.

Bei den Aktionen 21 und 29 zeigen sich Abweichungen zwischen den lokal erfassten Daten und den Cloud-Daten: Während Aktion 21 das Entriegeln des Schlosses durch einen Gastnutzer betrifft und Aktion 29 die Entriegelung durch den Hauptnutzer, weisen beide Aktionen die Problematik auf, dass im Protokoll jeweils eine andere Kommunikationsart dokumentiert wird als in der Cloud. Konkret wurde die Aktion in den Cloud-Daten als Fernsteuerung gekennzeichnet, während sie im selbst erstellten Protokoll als BLE-Vorgang erfasst wurde. Als durchführende Person wurde möglicherweise nicht berücksichtigt, dass sich das benutzte

Smartphone während der Ausführung der Aktion in einem anderen Netzwerk befand. Diese Diskrepanz stellt jedoch kein Sicherheitsproblem dar, da die korrekte Durchführung der Aktion in der Cloud zuverlässig erfasst wurde.

Die fehlenden Einträge der Aktionen 24 bis 26 lassen vermuten, dass die Umsetzung von Berechtigungsänderungen im System nicht sofort wirksam wurde und dadurch Verzögerungen entstanden. Dies kann erklären, warum Nutzer kurzzeitig trotz entzogener Rechte weiterhin das Schloss betätigen konnten.

Die nicht protokollierte Aktion 31, bei der das Schloss aus der Ferne verriegelt wurde, weist auf eine potenzielle Lücke in der Protokollierung hin. Sicherheitsrelevante Vorgänge werden in solchen Fällen möglicherweise nicht vollständig dokumentiert, was die Vollständigkeit der Ereignisdaten einschränkt.

Die Einträge, welche das Ent- und Verriegeln des Schlosses aus der Ferne dokumentieren, sind nicht ausreichend detailliert festgehalten. Zwar geht aus den Protokolleinträgen hervor, dass der Nutzer das Schloss remote, also aus der Ferne, betätigt hat, jedoch werden weder eine geografische Zone (Umkreis) noch der exakte Standort, beispielsweise in Form von Koordinaten, angegeben.

Im Rahmen der Untersuchungen ist festzustellen, dass eine Aktion bereits dann als fernsteuerungsbasiert erkannt wird, wenn sich das betreffende Endgerät nicht im Heimnetzwerk befindet. Somit wäre es beispielsweise möglich, dass der Hauptnutzer mit dem WLAN-Netzwerk eines Nachbarn verbunden ist und — während er sich noch in dessen Reichweite befindet — die Tür entriegelt. In einem solchen Fall würde im App-Protokoll dennoch der Eintrag „Aus der Ferne ent-/verriegelt“ erzeugt, obwohl sich das Endgerät physisch in unmittelbarer Nähe des Schlosses befindet.

Die Aktion 34 zeigt, dass die manuelle Betätigung des Schlosses über den Drehknauf in der Cloud in mehrere Einträge aufgeteilt wird – in diesem Fall zwei Einträge für das manuelle Entriegeln und zwei für das manuelle Verriegeln. Dies deutet darauf hin, dass die Cloud die einzelnen Betätigungsvorgänge auf einer feineren Ebene dokumentiert, als das im selbst erstellten Protokoll der Fall ist.

Insgesamt zeigen diese Beobachtungen, dass sowohl technische Implementierungen als auch Nutzungskontexte Einfluss auf die erfassten Daten haben. Diskrepanzen zwischen lokalen und Cloud-Daten sowie Verzögerungen bei Berechtigungsänderungen sind zentrale Faktoren, die bei der Interpretation der Ergebnisse berücksichtigt werden müssen.

### 5.1.3 App-Daten

Die mithilfe der Android Debug Bridge vom Smartphone des Hauptnutzers extrahierten App-spezifischen Daten erfordern eine vertiefte analytische Einordnung. Anhand der generierten Log-Dateien wird ersichtlich, dass die App eine aktive Protokollierung durchführt. Da jedoch keine der in den übrigen App-spezifischen Dateien identifizierten Session-IDs in den

Log-Dateien enthalten ist, kann keine eindeutige Zuordnung einer Session-ID zu einer konkreten Sitzung vorgenommen werden. Aufgrund fehlender öffentlicher Dokumentationen des Herstellers sowie dem Mangel an vergleichbaren Untersuchungen zu anderen Smart Lock Modellen ist eine verlässliche Bestimmung der jeweiligen Funktionalität der einzelnen Session-IDs nicht möglich.

Das Auftreten des BLE-Verbindungsstatus in den Log-Dateien weist darauf hin, dass das Ein- und Ausschalten der Bluetooth-Funktion im Zusammenhang mit den im Bluetooth-HCI-Snoop-Protokoll dokumentierten BLE-Kommunikationsvorgängen erfasst wird. Daraus kann geschlossen werden, dass nach Abschluss der Wiederherstellung der Bluetooth-Verbindung die jeweilige Aktion stattfindet. Nach der Erstellung des Smart Alerts „Gerät wird von einem bestimmten Benutzer ent- oder verriegelt“ und dessen Auslösung durch das Öffnen des Schlosses durch den Gastnutzer (Aktion 21) erfolgte die Generierung einer Push-Benachrichtigung, die auf dem Smartphone des Hauptnutzers angezeigt wurde. Für das Verriegeln des Schlosses durch den Gastnutzer (Aktion 22) konnte hingegen keine Push-Benachrichtigung festgestellt werden. Eine mögliche Ursache hierfür liegt darin, dass der Hauptnutzer während des Entriegelungsvorgangs die App nicht aktiv verwendete, während er beim Verriegelungsvorgang innerhalb der App aktiv war.

Die Analyse der Datenbankeinträge zeigt, dass jeweils nur die erste ausgeführte Aktion einer bestimmten Thematik in der entsprechenden Datenbank gespeichert wird. In der Tabelle *RuleData* der Datenbank *ModelDatabase.db* ist demnach ausschließlich die erste Zugangsberechtigungsregel (Aktion 4) hinterlegt, ebenso wie in der Tabelle *UserLocation* der Datenbank *LocationDatabase.db* lediglich die Aktivierung der Auto Unlock Funktion verzeichnet ist. In der Tabelle *UserData* der Datenbank *ModelDatabase.db* wurden für den Gastnutzer zwei Einträge angelegt. Dies ist darauf zurückzuführen, dass der Hauptnutzer beim Versenden der Einladung an den Gastnutzer aufgefordert wurde, dessen Vor- und Nachnamen sowie die Telefonnummer anzugeben (siehe Tabelle 4.4 User Data 2). Die Anmeldung des Gastnutzers in der App erfolgte über dessen Google-Konto (siehe Tabelle 4.4 User Data 3). Dieser Umstand erklärt das Vorhandensein von zwei Einträgen.

In der Datenbank *ModelDatabase.db* in der Tabelle *UserData* wurden für den Gastnutzer zwei Einträge erstellt. Das liegt daran, dass während dem Versenden der Einladung an den Gastnutzer, der Hauptnutzer dazu aufgefordert wurde, den Vor- und Nachnamen sowie die Telefonnummer des zu Berechtigenden anzugeben. Das erklärt den zweiten Eintrag (siehe Tabelle 4.4 User Data 2). Die Anmeldung des Gastnutzers in der App erfolgte mithilfe dessen Google Kontos (siehe Tabelle 4.4 User Data 3).

### 5.1.4 Manueller Vergleich der Ereignisprotokolle

Die Analyseergebnisse weisen eine deutliche Nähe zu den in der Cloud erfassten Daten auf, wodurch auch die zugrunde liegenden Begründungen in weiten Teilen übereinstimmen. Dies zeigt, dass die wichtigsten Beobachtungen durch mehrere Datenquellen gestützt werden.

Eine wesentliche Erkenntnis betrifft den Umgang des Systems mit Nutzerinteraktionen, die keinen sicherheitskritischen Charakter aufweisen. Das bloße Anzeigen oder Aktualisieren von Einträgen durch den Hauptnutzer führt zu keinen zusätzlichen Protokollierungen. Dies legt nahe, dass derartige Vorgänge vom System bewusst gefiltert werden, um eine Überlastung des Protokolls mit irrelevanten Informationen zu vermeiden und den Fokus auf sicherheitsrelevante Ereignisse zu lenken.

Problematisch ist hingegen das Ausbleiben einer Protokollierung eines unzulässigen Zugriffsversuchs durch einen Gastnutzer außerhalb seines Berechtigungszeitraums. Das Fehlen eines solchen Eintrags erschwert nicht nur die Nachvollziehbarkeit potenzieller Angriffe, sondern wirft auch Fragen hinsichtlich der Sensitivität des Systems bei der Erkennung missbräuchlicher Zugriffsvorgänge auf.

Entsprechend der im Unterabschnitt 5.1.2 getroffenen Feststellungen ist das Nichtvorhandensein eines Eintrags zur Erstellung eines Smart Alerts (Aktion 20) auch im vorliegenden Zusammenhang als Einschränkung der Nachvollziehbarkeit zu bewerten.

Darüber hinaus zeigen sich bei einzelnen Aktionen Unterschiede zwischen lokal erfassten Daten und den Aufzeichnungen des App-Ereignisprotokolls. Insbesondere bei den Aktionen 21 (Entriegelung durch einen Gastnutzer) und 29 (Entriegelung durch den Hauptnutzer) fällt auf, dass unterschiedliche Kommunikationsarten dokumentiert werden. Während das App-Protokoll diese Ereignisse als Fernsteuerungen ausweist, werden diese im händisch erstellten Protokoll als BLE-Vorgänge registriert. Eine plausible Erklärung für diese Abweichung liegt in der Netzwerkumgebung des verwendeten Endgeräts während der Ausführung. Aus sicherheitstechnischer Perspektive hat diese Differenz jedoch keine unmittelbare Relevanz, da die Aktionen im App-Protokoll dokumentiert sind.

Die nicht erfassten Aktionen 24 bis 26 geben Hinweise auf Verzögerungen bei der Umsetzung von Berechtigungsänderungen. Es ist anzunehmen, dass Änderungen an den Zugriffsrechten nicht sofort systemweit wirksam wurden, wodurch Nutzer kurzfristig trotz entzogener Berechtigungen weiterhin Zugriff hatten. Dieses Verhalten deutet auf mögliche Latenzen in der Synchronisation zwischen den Systemkomponenten hin und offenbart Optimierungsbedarf im Berechtigungsmanagement.

Ergänzend fällt ein zusätzliches, nicht eindeutig erklärbares Entriegeln aus der Ferne durch den Hauptnutzer auf. Ob es sich hierbei um eine nachträgliche Übermittlung oder um einen automatisierten Systemprozess handelt, bleibt offen und sollte weitergehend untersucht werden, um Fehlinterpretationen vorzubeugen. In Anlehnung an die in Unterabschnitt 5.1.2 beschriebenen Feststellungen zur unzureichenden Dokumentation der Fernsteuerungsereignisse zeigt sich auch hier eine eingeschränkte Nachvollziehbarkeit der tatsächlichen Nutzungssituation.

Auch die detaillierte Darstellung physischer Interaktionen im App-Ereignisprotokoll ist hervorzuheben. Am Beispiel von Aktion 34 zeigt sich, dass die manuelle Betätigung des Schlosses über den Drehknauf in mehrere Einträge zerlegt wird. Diese feingranulare Erfassung ermöglicht eine präzise Rekonstruktion einzelner Handlungen.

Die geringfügigen Abweichungen in der Dokumentation der Zeitstempel lassen sich dadurch erklären, dass das System die Ausführung einer Aktion nicht unmittelbar beim Betätigen des Buttons gewährleistet, sondern hierfür eine gewisse Bearbeitungszeit benötigt. Dadurch kann es vorkommen, dass die Ausführung bereits in die folgende Minute fällt und diese im Protokoll notiert wird.

Die Ergebnisse zeigen, dass sowohl technische Faktoren wie Netzwerk und Synchronisation als auch die konkrete Nutzung der App die erfassten Daten stark beeinflussen. Unterschiede zwischen den Protokollen sowie Verzögerungen bei Berechtigungen sind daher wichtige Punkte, die bei der Analyse berücksichtigt werden müssen, um verlässliche Schlüsse zu ziehen.

## 5.2 Forensische Relevanz

Die forensische Relevanz von Smart Locks ergibt sich nicht nur aus einer einzelnen Datenquelle, sondern aus dem Zusammenspiel verschiedener Informationen. Erst durch die Verknüpfung der unterschiedlichen Artefakte entsteht ein konsistentes Gesamtbild. Einzelne Teilaspekte ergänzen sich gegenseitig und ermöglichen so eine umfassendere Bewertung.

Von besonderem Wert ist das Ereignisprotokoll der App, da es eine unmittelbare Verbindung zwischen durchgeführter Aktion, Zeitpunkt und verantwortlicher Person herstellt. So lassen sich sicherheitsrelevante Vorfälle, etwa Einbrüche oder Diebstähle, mit aufgezeichneten Einträgen abgleichen. Auf diese Weise können Personen entweder als mögliche Täter in Betracht gezogen oder ausgeschlossen werden. Darüber hinaus wird dokumentiert, ob ein Zugriff lokal über BLE oder aus der Ferne erfolgte. Dies ermöglicht prinzipiell die Feststellung physischer Anwesenheit. Allerdings ist hier ein methodisches Problem zu berücksichtigen: Sobald ein Smartphone über ein anderes Netzwerk als das hinterlegte Heimnetzwerk eingebunden wird, wird die Aktion als Fernzugriff vermerkt. Befindet sich dieses alternative Netzwerk in unmittelbarer Nähe des Wohnortes, verliert die Unterscheidung zwischen physischer Präsenz und Abwesenheit deutlich an Aussagekraft.

Zudem werden Veränderungen an den Zugriffsrechten festgehalten. Das Hinzufügen oder Entfernen von Berechtigungen, sei es dauerhaft, temporär, wiederkehrend oder mit dem Status Eigentümer verbunden, dokumentiert die Rollenvergabe und deren zeitlichen Rahmen. Allerdings ist nicht nachvollziehbar, welche konkrete Art der Änderung vorgenommen wurde. Lediglich der Zeitraum, in dem eine Berechtigung bestand, ist verlässlich nachvollziehbar.

Auch die Verwaltung von Smart Alerts ist nur eingeschränkt erfasst. Zwar ließe sich aus deren Aktivierung oder Deaktivierung ableiten, ob ein Nutzer mit erweiterten Rechten bspw. die Benachrichtigungsfunktion gezielt außer Kraft gesetzt hat, um die Überwachung zu umgehen. Da jedoch entsprechende Einträge fehlen, bleibt ein möglicher Hinweis auf vorbereitete Manipulationen, etwa vor einem Einbruch, ungenutzt.

Die Identifizierbarkeit der Nutzer hängt maßgeblich von der Genauigkeit der gespeicherten Kennungen ab. Allgemeine Bezeichnungen wie „Google User“ besitzen nur geringe Aussage- bzw. Beweiskraft. Eindeutige Identifikatoren wie vollständige Namen, Gmail-Adressen oder

Telefonnummern erhöhen dagegen den forensischen Wert erheblich, da sie eine zweifelsfreie Zuordnung zu einer Person ermöglichen. Dennoch ist stets zu berücksichtigen, dass ein berechtigter Zugang nicht zwingend bedeutet, dass die betreffende Person selbst gehandelt hat. So könnte etwa ein entwendetes Smartphone missbraucht oder ein Benutzerkonto kompromittiert worden sein. In solchen Fällen wäre eine Handlung im Protokoll zwar formal korrekt zugeordnet, tatsächlich jedoch von einer unbefugten Person ausgeführt worden. Um dennoch Klarheit zu gewinnen, kann die vom Smartphone-Diebstahl betroffene Person eine Anzeige bei der Polizei erstatten. Diese Meldung lässt sich zeitlich mit dem Zugriff über das entwendete Gerät abgleichen, sodass der vermeintliche Täter ausgeschlossen werden kann. Wie bereits in Abschnitt Abschnitt 2.5 „Telefon verloren“ erläutert, besteht darüber hinaus die Möglichkeit, den in der yale Home App hinterlegten Account zu sperren. Auf diese Weise kann verhindert werden, dass das zuvor beschriebene Szenario überhaupt eintritt und ein unbefugter Zugriff auf das Benutzerkonto erfolgt.

Darüber hinaus werden Daten zusätzlich extern gesichert, wodurch eine Redundanz entsteht, die vor Löschung oder Manipulation auf Endgeräten schützt. Allerdings weist auch diese Form der Speicherung Lücken auf, die bereits im Ereignisprotokoll der App erkennbar sind. Zusätzlich werden dort Kontaktdaten wie Telefonnummern und E-Mail-Adressen abgelegt, was die Identifizierung einzelner Mitnutzer und deren Zugriffsrechte erleichtert. In einer weiteren Spalte wird zudem die Zeitzone dokumentiert, was Rückschlüsse auf den Aufenthaltsort bei der Ausführung einer Aktion zulassen könnte. Ob diese Information tatsächlich geeignet ist, den Standort zuverlässig zu bestimmen, bedarf jedoch einer vertieften Untersuchung (siehe Abschnitt 5.4). Sollte sich dies bestätigen, ließe sich damit auch das zuvor beschriebene Problem, ob eine Aktion wirklich aus der Ferne erfolgte, präzise klären.

Insgesamt können die in der Cloud gesicherten Daten aufgrund ihrer geringen Manipulierbarkeit und direkten Nutzerzuordnung als besonders beweiskräftig eingestuft werden. Sie besitzen damit ein hohes Potenzial, in gerichtlichen Verfahren als belastbares Beweismittel herangezogen zu werden.

## 5.3 Limitationen und Herausforderungen

Neben den dargestellten Ergebnissen ist es für die Einordnung der Arbeit von wesentlicher Bedeutung, deren Grenzen kritisch zu reflektieren. Die Erhebung der Datenquellen im Bereich Netzwerk, Cloud und App sowie der Vergleich der Ereignisprotokolle des Yale Linus® Smart Locks bringen dabei verschiedene Limitationen und Herausforderungen mit sich, die sowohl die theoretische Ausrichtung der Untersuchung als auch methodische und interpretative Aspekte betreffen.

Bezüglich der theoretischen Limitationen ist zu beachten, dass sich die Datenerhebung ausschließlich auf das Yale Linus® Smart Lock und die zugehörige Yale Home App konzentriert. Andere Smart Lock-Modelle desselben Herstellers sowie Produkte anderer Anbieter wurden nicht berücksichtigt. Zudem erfolgte die Steuerung der Aktionen über ein gerootetes Smartphone mit dem Betriebssystem Android 14, sodass andere Android-Versionen sowie

alternative Betriebssysteme wie iOS außen vor blieben. Hinzu kommt, dass lediglich die Firmware-Version 3.2.2 des Schlosses getestet wurde, wodurch mögliche Abweichungen bei anderen Versionen nicht ausgeschlossen werden können.

Darüber hinaus wurden im Untersuchungsdesign nicht alle verfügbaren Funktionen des Schlosses berücksichtigt. Zwar standen die zentralen Funktionen im Fokus, einige wenige Aspekte wurden jedoch außer Acht gelassen, sodass unklar bleibt, welche Systemeinträge diese auslösen würden. Dazu gehören beispielsweise die DoorSense™-Technologie, die Steuerung des Schlosses über eine Smartwatch oder einen Sprachassistenten sowie ein Teil der verfügbaren Smart Alerts. Auch die Funktion „Auto Unlock“ konnte nicht analysiert werden, da sie im Versuch nicht wie vorgesehen funktionierte.

Methodische Limitationen ergeben sich vor allem daraus, dass es lediglich möglich war, den BLE-Verkehr zwischen Smartphone und Schloss detaillierter zu untersuchen. Der Netzwerkverkehr war hingegen verschlüsselt und bot daher kaum verwertbare Informationen. Auch in Bezug auf den BLE-Traffic konnten nur Korrelationen festgestellt werden, da öffentliche Dokumentationen zu herstellerspezifischen UUIDs oder proprietären Protokollen fehlen. Diese Lücken erschwerten das Analyseverfahren erheblich, da die Ergebnisse stark von den erhobenen Daten abhängig sind und durch die Verschlüsselung keine kausale Auswertung möglich war.

Interpretative Limitationen ergeben sich schließlich daraus, dass die eigenen Schlussfolgerungen ebenfalls Grenzen aufweisen. Aufgrund fehlender offizieller Dokumentationen sowie nicht verfügbarer Angaben zu herstellerspezifischen Werten wie UUIDs konnten ausschließlich Korrelationen festgestellt, jedoch keine Kausalitäten identifiziert werden. So beziehen sich die Annahmen zu einzelnen Handles (z. B. für Authentifizierung, Statusbestätigung oder Aktionssteuerung) auf beobachtete Pakete, deren Aufteilung durch die Datenmenge bedingt war. Diese Ergebnisse sind daher eher als Hypothesen, als gesicherte Aussagen zu bewerten.

Zudem war das Schloss bereits vor der Datenerhebung eingerichtet, sodass zu diesem Zeitpunkt bereits eine BLE-Verbindung zwischen dem Smartphone und dem Schloss bestand. Aus diesem Grund konnten keine SMP-Pakete mitgeschnitten bzw. aufgezeichnet werden, wodurch es nicht möglich war, das während des Pairings verwendete Association Model zu bestimmen.

Auch der Vergleich des eigens erstellten Protokolls mit dem App-Protokoll offenbarte Lücken und Ungenauigkeiten. Bestimmte Aktionen wurden darin nicht dokumentiert, beispielsweise das Entriegeln der Tür durch einen Gastnutzer außerhalb zulässiger Zeiträume oder die Einrichtung von Smart Alerts. Zudem wies der Zeitabgleich Abweichungen von bis zu einer Minute auf, was – wie in Unterabschnitt 5.1.4 erläutert – durch Bearbeitungszeiten im System erklärbar ist. Weiterhin erschwert die eindeutige Zuordnung von Gastnutzern die Nachvollziehbarkeit: Bei der Anmeldung über ein Google-Konto erscheinen Nutzer lediglich als „Google User“, ohne individuelle Differenzierung. Dies wirft Fragen auf, insbesondere, wenn mehrere Personen dasselbe Registrierungsverfahren nutzen.

Aus diesen Limitationen ergeben sich mehrere zentrale Herausforderungen. So dokumentiert das Ereignisprotokoll lediglich das verwendete Endgerät, nicht jedoch die tatsächlich handelnde Person, was insbesondere bei einem gestohlenen Smartphone problematisch sein kann. Der überwiegende Teil der Daten ist verschlüsselt, auch nach Erlangung von Root-Rechten und Zugriff auf das Dateisystem. Langzeittests konnten nicht durchgeführt werden, da das Schloss in einer Versuchsumgebung und nicht im alltäglichen Gebrauch integriert war. Zudem fehlen öffentliche Dokumentationen seitens Yale, August oder anderer Hersteller, sodass ein Abgleich der Daten, etwa in Bezug auf herstellerspezifische UUIDs für die BLE-Analyse, nicht möglich war. Schließlich standen nur wenige externe Quellen zur Verfügung, deren Nutzen für die Auswertung begrenzt blieb. Unklar ist beispielsweise die Bedeutung bestimmter App-spezifische Daten wie verschiedener Session-IDs.

## 5.4 Ausblick

Im Rahmen dieser Arbeit konnten zentrale Aspekte der Funktionsweise und Kompatibilität des untersuchten Smart Locks aufgezeigt werden. Dennoch ergeben sich aus den Ergebnissen verschiedene Fragestellungen, die in zukünftigen Untersuchungen vertieft werden sollten. Ein naheliegender Ansatzpunkt ist die Prüfung der Kompatibilität mit einer größeren Bandbreite an Endgeräten. Da sich die vorliegenden Tests auf eine begrenzte Auswahl stützten, wäre es sinnvoll, Smartphones unterschiedlicher Betriebssysteme sowie verschiedener Betriebssystemversionen einzubeziehen. Auf diese Weise ließe sich feststellen, ob die beobachtete Funktionsweise konsistent bleibt oder ob es Unterschiede in Zuverlässigkeit und Nutzererlebnis gibt.

Ebenso denkbar ist eine Analyse der verwendeten Softwareumgebung. Unterschiedliche Versionen der Yale Home App könnten sich in der Bedienbarkeit, in den verfügbaren Funktionen oder auch in der Zuverlässigkeit des Systems bemerkbar machen. Hier könnte eine systematische Untersuchung aufzeigen, welche Rolle die jeweilige App-Version für die Gesamtfunktionalität spielt.

Darüber hinaus stellt sich die Frage, inwieweit die Ergebnisse über das konkrete Modell hinaus übertragbar sind. Die Einbeziehung weiterer Produkte aus der Yale-Reihe sowie von Smart Locks anderer Hersteller wäre erforderlich, um die Generalisierbarkeit der Resultate zu prüfen. Nur so lässt sich klären, ob die gewonnenen Erkenntnisse spezifisch für das untersuchte Schloss gelten oder ob sie Rückschlüsse auf ein breiteres Spektrum an Smart Lock-Systemen zulassen.

Neben der Hardware- und Softwarevielfalt bietet auch der Funktionsumfang Ansatzpunkte für weiterführende Analysen. Insbesondere die Frage, wie zusätzliche Funktionen dokumentiert und protokolliert werden, könnte wertvolle Hinweise auf die Nachvollziehbarkeit und Sicherheit der Nutzung liefern. Hier wäre es interessant, nicht nur die technische Implementierung, sondern auch die praktische Anwendbarkeit im Alltag zu beleuchten.

Darüber hinaus könnte ein längerer Testzeitraum neue Erkenntnisse ermöglichen. Während die vorliegende Untersuchung in einer eher kontrollierten Umgebung stattfand, wäre ein Praxistest über mehrere Wochen oder Monate hinweg sinnvoll, um Aspekte wie Zuverlässigkeit, Wartungsaufwand und Nutzerfreundlichkeit im Alltag zu evaluieren.

Um das während des Pairings verwendete Association Model ermitteln zu können, wäre es erforderlich, die bestehende Bluetooth-Verbindung zwischen dem Smartphone und dem Schloss vollständig zu trennen oder ein anderes Smartphone zu verwenden, welches noch keine Verbindung hergestellt hat, um dadurch ein neues BLE-Pairing zu initiieren.

Schließlich sollte die Verwaltung mehrerer Benutzerzugänge eingehender betrachtet werden. Szenarien, in denen mehrere Gastnutzer über Google-Mail-Konten hinzugefügt werden, werfen Fragen hinsichtlich der internen Verwaltung und der Eindeutigkeit der Zuordnung auf. Hier könnte beispielsweise untersucht werden, ob das System identische Namen vergibt oder diese durch Zusätze wie Ziffern unterscheidet. Solche Untersuchungen wären nicht nur aus technischer Sicht relevant, sondern auch für die Benutzerfreundlichkeit und Transparenz des Systems von Bedeutung.

Es bleibt offen, in welchem Umfang Gastnutzer nach einer Rollenänderung agieren können: Wenn der ursprüngliche Hauptbenutzer einen Gastnutzer zum Eigentümer macht, stellt sich die Frage, ob dieser neue Eigentümer seinerseits weitere Gastnutzer hinzufügen kann. Ebenso ist zu prüfen, ob Gastnutzer – unabhängig von einer Eigentümerrolle – Smart Alerts erstellen können oder ob diese Funktion ausschließlich Eigentümern vorbehalten ist. Weiter ist zu klären, ob der ursprüngliche Hauptnutzer durch einen zum Eigentümer ernannten Gastnutzer entfernt werden kann, was direkte Auswirkungen auf die Berechtigungsverwaltung hätte.

Darüber hinaus ist zu berücksichtigen, wie sich der Zugriff auf das System aus unterschiedlichen Zeitzonen auswirkt. Es stellt sich die Frage, ob bei der Steuerung des Schlosses von einem Telefon außerhalb der eigenen Zeitzone die abweichende Zeitzone korrekt in den Protokollen vermerkt wird oder ob weiterhin die lokale Zeitzone (z. B. Berlin/Europe) angezeigt wird. Dies ist relevant für die Sicherheit, Nachvollziehbarkeit von Zugriffen und die Konsistenz der Protokolldaten.

Im Rahmen zukünftiger Untersuchungen sollte auch geklärt werden, warum in den Tabelle *RuleData* der Datenbank *ModelDatabase.db* sowie in der Tabelle *UserLocation* der Datenbank *LocationDatabase.db* ausschließlich die initiale Aktion gespeichert wird und fortlaufende Aktionen keine Berücksichtigung in der jeweiligen Datenbank finden.

## 5.5 Fazit

Die Ergebnisse dieser Arbeit knüpfen an die Untersuchung des August Smart Lock Pro durch Apthorpe et al. an [16]. Dies verdeutlicht die Relevanz vergleichender Analysen unterschiedlicher Smart Lock Modelle, um ein tieferes Verständnis für die Funktionsweise, Datenspeiche-

rung und -verarbeitung vernetzter Geräte zu entwickeln. Vernetzte IoT-Systeme stellen eine Vielzahl digitaler Spuren bereit, die in forensischen Analysen eine wesentliche Rolle bei der Aufklärung von sicherheitsrelevanten und strafrechtlich bedeutsamen Vorfällen spielen.

Die vorliegende Arbeit untersuchte das Yale Linus® Smart Lock sowie die dazugehörige Yale Home App eingehend. Die Analyse lieferte eine Vielzahl forensisch relevanter Erkenntnisse. Neben der Untersuchung der Funktionalität des Geräts und der App konnten auch die Verarbeitung und Speicherung von Nutzerdaten nachvollzogen sowie sicherheitsrelevante Aspekte betrachtet werden. Ein Teilziel dieser Arbeit bestand darin, die während der Nutzung eines Yale Linus® Smart Locks entstehenden digitalen Spuren zu identifizieren. Hierfür wurde ein mehrstufiges methodisches Vorgehen gewählt.

Um forensisch verwertbare Erkenntnisse zu gewinnen, kam für jede Datenquelle eine angepasste Methodik zur Datenerhebung zum Einsatz. Die Erfassung der Netzwerkdaten und des BLE-Verkehrs erfolgte mithilfe der Open-Source-Software OPNSense sowie des Bluetooth-HCI-Snoop-Protokolls. Für den Zugriff auf in der Cloud gespeicherte Daten wurde über die E-Mail-Adresse <*support@yalehome.de*> eine Anfrage an den Anbieter gestellt, woraufhin die relevanten Datensätze heruntergeladen werden konnten. Die Gerätedaten des Hauptnutzers wurden über die Android Debug Bridge extrahiert. Zusätzlich wurde während der Datenerhebung ein eigenes Protokoll geführt, das mit dem App-Ereignisprotokoll abgeglichen werden konnte. Dabei zeigte sich, dass auch ein Abgleich mit weiteren Datenquellen möglich war.

Ziel dieser Vorgehensweise war es, zu ermitteln, welche Netzwerk-, Cloud- und App-Daten forensisch von Bedeutung sind und welchen Beitrag sie zur Untersuchung sicherheitsrelevanter Vorfälle leisten. Zudem sollte festgestellt werden, welche Informationen sich aus der zugehörigen Yale Home App extrahieren lassen, über die die Steuerung der Aktionen erfolgt.

Die Netzwerkanalyse ergab, dass das Schloss ausschließlich verschlüsselte Verbindungen nutzt, basierend auf der TLS-Protokoll Version 1.2. Bei der Untersuchung der Metadaten konnten Zieladressen der Kommunikation über die Bridge identifiziert werden. Die IP-Adressen 3.33.245.2 und 15.197.218.233 gehören zu Servern der Amazon Technologies Inc., die als Cloud-Server zur Speicherung von Nutzeraktivitäten dienen und eine Synchronisation mit dem in der App aufgezeichneten Ereignisprotokoll ermöglichen. Darüber hinaus konnten sowohl die LAN- als auch die WAN-MAC-Adresse bestimmt werden, welche die Kommunikation innerhalb bzw. außerhalb des lokalen Netzwerks abbilden.

Die Analyse des BLE-Verkehrs lieferte Erkenntnisse zum verwendeten LE 1M PHY-Modus der Datenübertragung sowie zu verschiedenen Parametern des Advertising-Prozesses. Zudem konnten die Gerätenamen und Bluetooth-Adressen der beteiligten Geräte (Google Pixel 7a und Yale Linus® Smart Lock) identifiziert werden. Auch die verwendeten Services und Characteristics ließen sich bestimmen. Eine präzisere Zuordnung der herstellerspezifischen, in Form von 128-Bit-UUIDs dargestellten Services und Characteristics war aufgrund fehlender öffentlicher Dokumentation nicht möglich. Es konnte jedoch festgestellt werden, dass das Handle 0x0017 der Authentifizierung dient, während 0x0019 deren Bestätigung übernimmt. Das Handle 0x0012 führt Aktionen aus, und 0x0014 bestätigt deren Ausführung. Jede Aktion wird somit über das gleiche Handle gesteuert.

Die Cloud-Daten lagen als CSV-Datei vor und enthielten Einträge zu sämtlichen ausgeführten Aktionen. Für jeden Eintrag wurde der Zeitstempel, die beschriebene Aktion, der ausführende Nutzer, die Telefonnummer, die Geräte-ID sowie die Lokalisation (in Form der Zeitzone) gespeichert. Für den Hauptnutzer wurde zusätzlich die hinterlegte E-Mail-Adresse vermerkt, während diese bei Gastnutzern entfällt. Bei automatisch oder manuell ausgelösten Aktionen fehlen Angaben zu Nutzer, Telefonnummer und E-Mail-Adresse vollständig.

Die Analyse der extrahierten App-Daten zeigte, dass das Auslösen von Smart Alerts Einträge in den Log-Dateien erzeugt und somit Rückschlüsse auf deren Aktivierung zulässt. In den Tabellen *BridgeData*, *HouseData* und *LockData* der Datenbank *ModelDatabase.db* sind Informationen über die Bridge, das Haus und das Schloss gespeichert, die eine eindeutige Zuordnung der verwendeten Geräte und deren Bezeichnungen ermöglichen. Die Tabelle *UserData* derselben Datenbank erlaubt eine eindeutige Nutzerzuordnung. Für den Gastnutzer konnten zwei Einträge identifiziert werden: Einer dokumentiert das Versenden der Einladung mit Angabe von Vor- und Nachname sowie Telefonnummer, der zweite das Anmelden des Gastnutzers über dessen Google-Konto.

Die Tabelle *RuleData* enthält Einträge zu den erstellten Zugangsberechtigungsregeln, während in der Tabelle *UserLocation* der Datenbank *LocationDatabase.db* festgehalten wird, für welche Tür und zu welchem Zeitpunkt die Auto-Un-/Lock-Funktion aktiviert bzw. deaktiviert wurde und welches Nutzerverhalten (z. B. Ankunft oder Verlassen des Hauses) vorlag. Eine Einschränkung besteht darin, dass in den Tabellen *RuleData* und *UserLocation* jeweils nur die erste ausgeführte Aktion protokolliert und gespeichert wird.

Der Vergleich des App-Protokolls mit dem selbst geführten Protokoll ergab, dass sämtliche Vorgänge des Ver- und Entriegelns durch Haupt- und Gastnutzer sowie das Gewähren von Zugangsrechten im Protokoll vermerkt sind. Nicht dokumentiert ist hingegen die Art der erteilten Zugangsberechtigung, die lediglich in der Gästeliste eingesehen werden kann, solange diese aktiv ist. Einstellungen wie das De-/Aktivieren der Auto Un-/Lock-Funktion oder das Erstellen eines Smart Alerts werden im App-Ereignisprotokoll nicht erfasst.

Die Untersuchung verdeutlichte, dass erst das Zusammenspiel der Artefakte aus verschiedenen Datenquellen ein vollständiges Gesamtbild liefert. Einzelne Datenquellen oder Artefakte allein ermöglichen keinen hinreichenden Rückschluss auf einen Vorfall. Die Artefakte ergänzen sich gegenseitig und sind in ihrer Gesamtheit für forensische Untersuchungen von erheblicher Relevanz und praktischer Anwendbarkeit.

Die Cloud-Daten und das App-Protokoll bilden die Grundlage zur Nachvollziehbarkeit der durchgeführten Nutzeraktionen. Ohne die in den Log-Dateien und Datenbankeinträgen identifizierten Artefakte wäre jedoch keine detaillierte Rekonstruktion der Benutzerinteraktionen möglich.

# Anhang A: Abbildungen

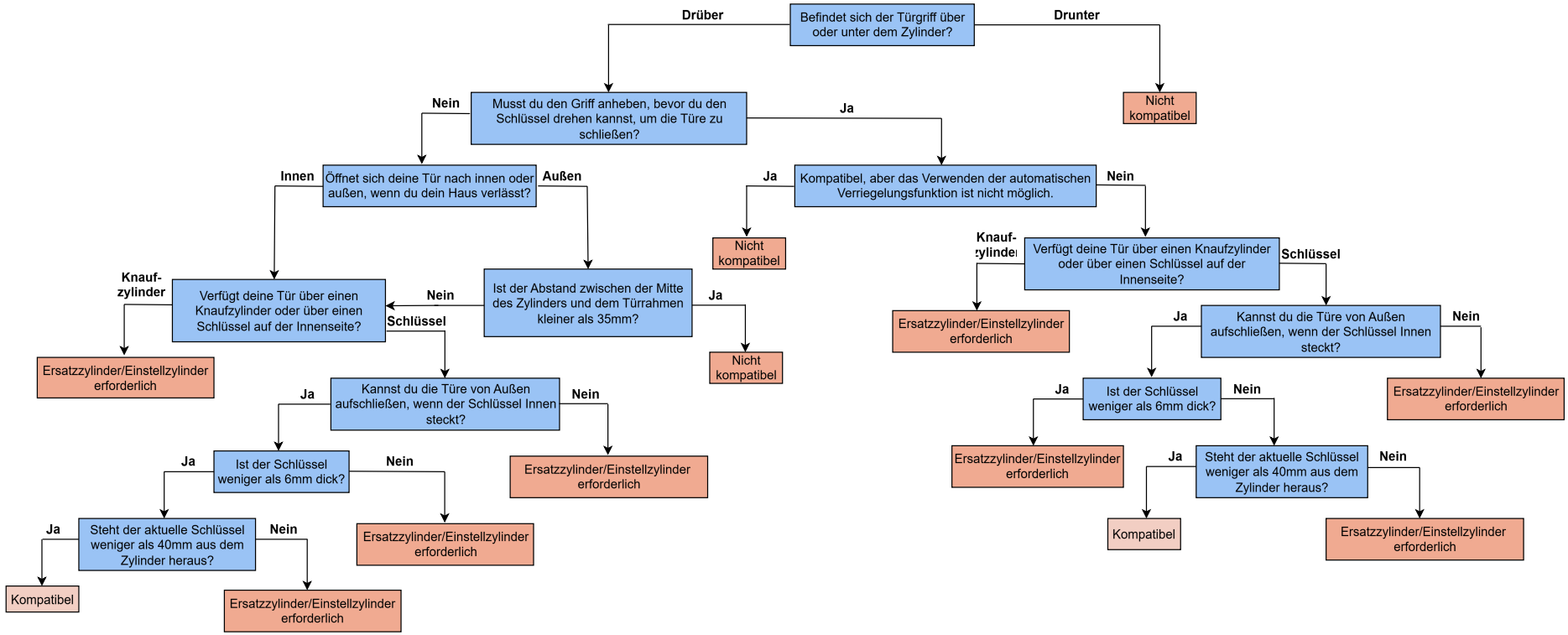
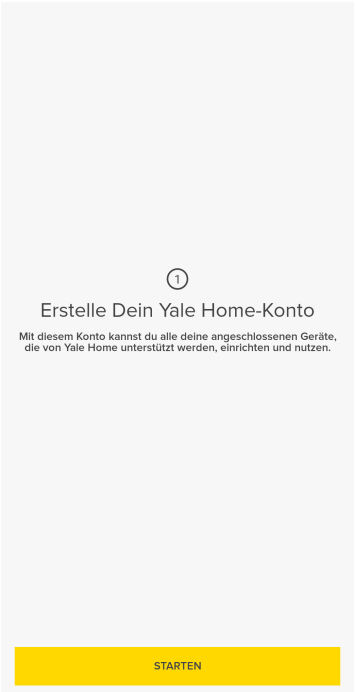
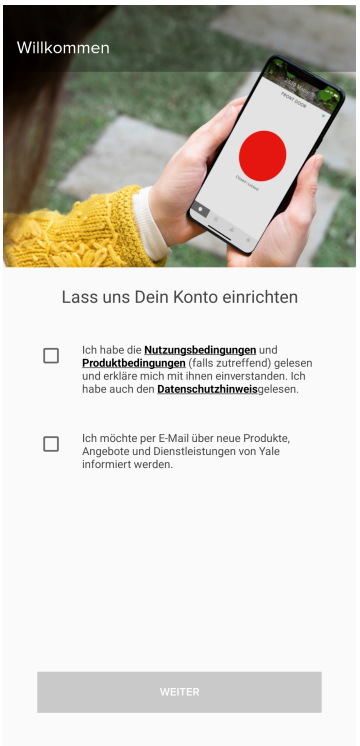


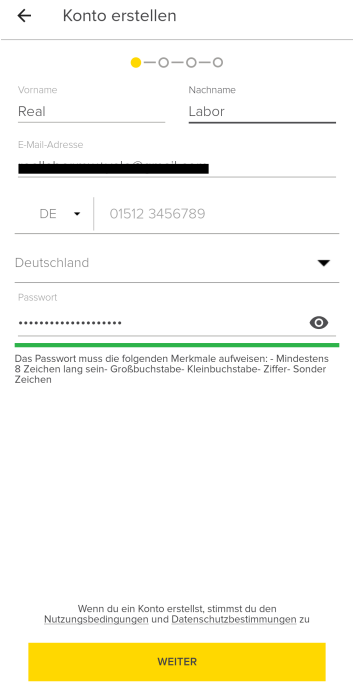

**Abbildung A.1:** Binärer Entscheidungsbaum zur Bestimmung der Kompatibilität des Zylinderschlosses [24]



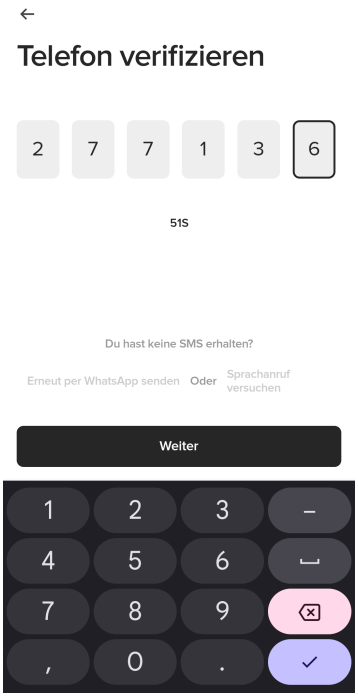

**(a)** Registrierung starten



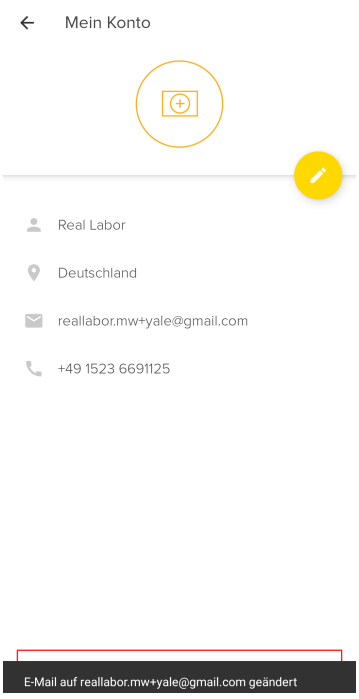

**(b)** Nutzungs- und Produktbedingungen und Datenschutzerklärung




**(c)** Dateneingabe



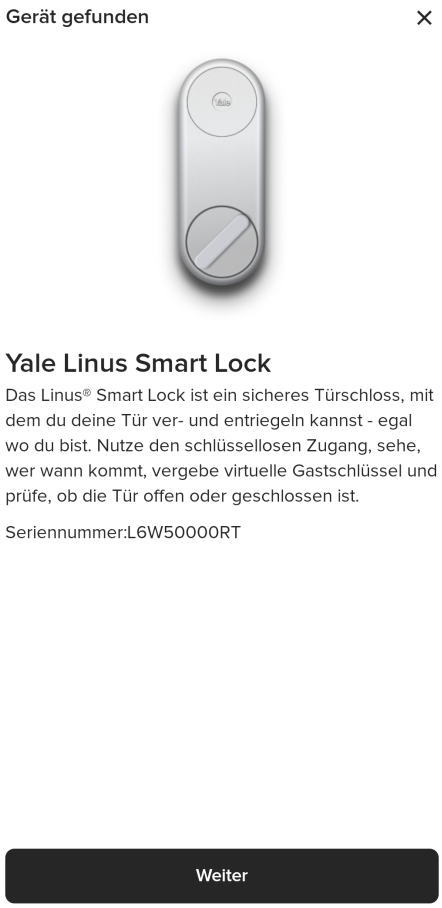

**(d)** Verifizierung




**(e)** Registrierung abgeschlossen

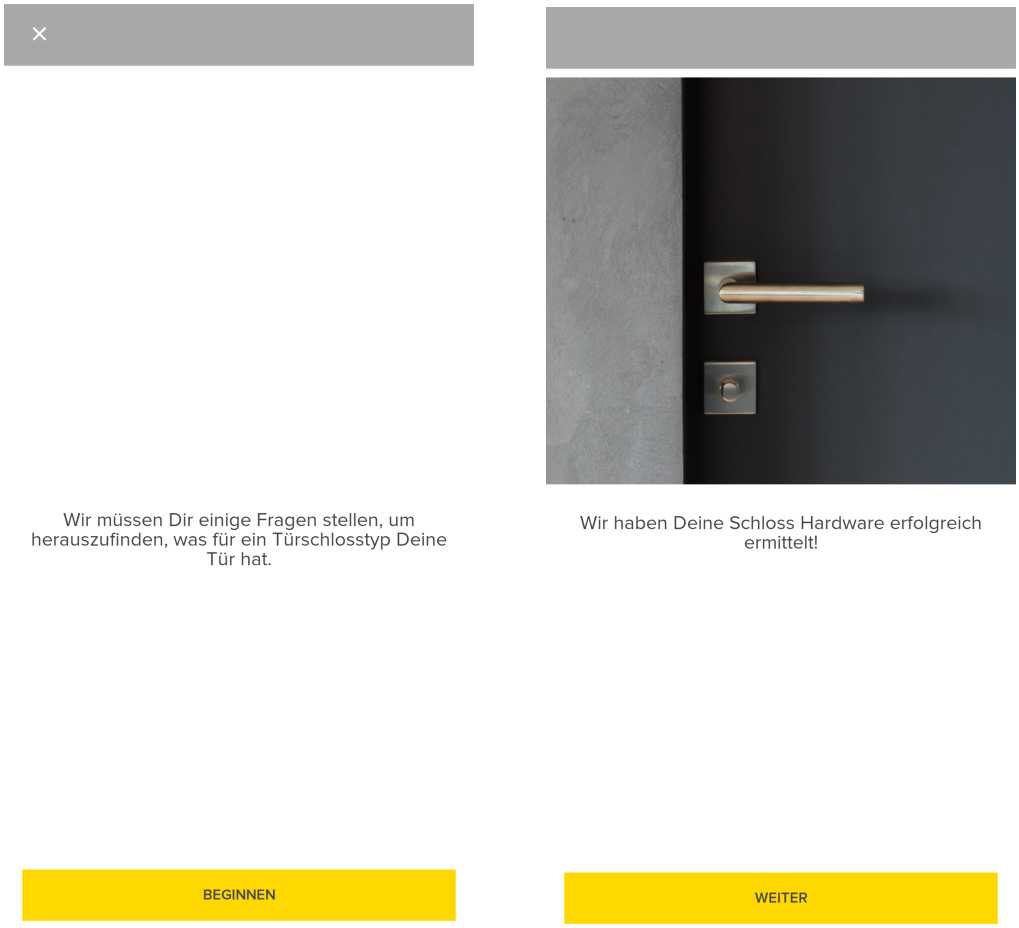
**Abbildung A.2:** Erstellung Yale Home-Konto [29]


**(a)** Hauptmenü



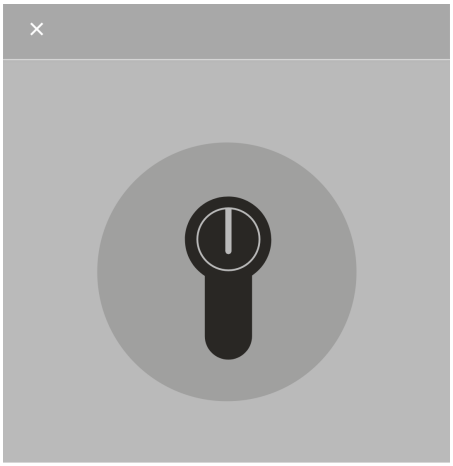

**(b)** Automatische Erkennung des Schlosses

**Abbildung A.3:** Einrichten eines neuen Geräts [29]




**(a)** Türschlossart **(b)** Hardware

**Abbildung A.4:** Erkennung der Türschlossart und Feststellung der Hardware [29]

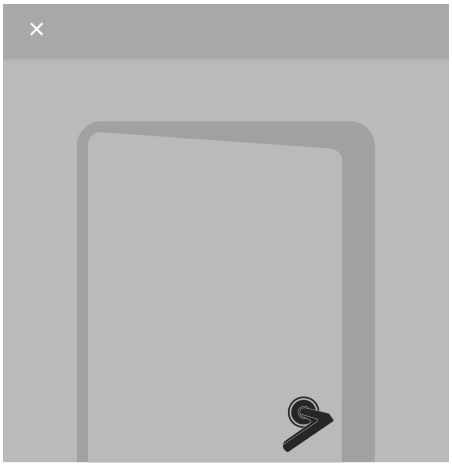


Hat Dein Schloss ein Europrofilzylinder?

JA

NEIN

**(a)** Europrofilzylinder

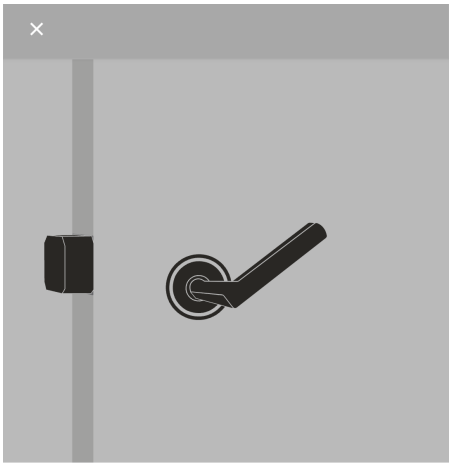


Hat Deine Tür einen Drücker, der die Falle von außen betätigt?

JA, ICH KANN DIE TÜR ÖFFNEN

NEIN, NUR EIN KNAUF ODER STOSSGRIFF

**(b)** Drücker



Muss der Türgriff zum Verriegeln hochgeklappt werden?

JA, ER MUSS HOCHGEKLAPPT WERDEN

NEIN, ICH KANN NORMAL VERRIEGELN

**(c)** Verriegelung

**Abbildung A.5:** Fragen zur Art des Türschlosses [29]

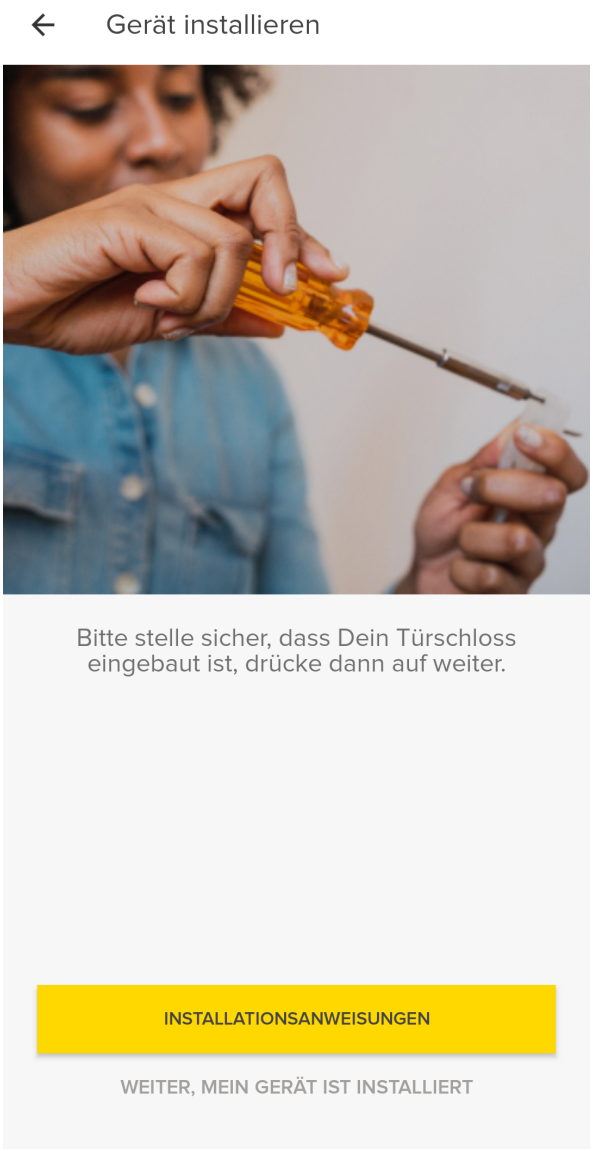
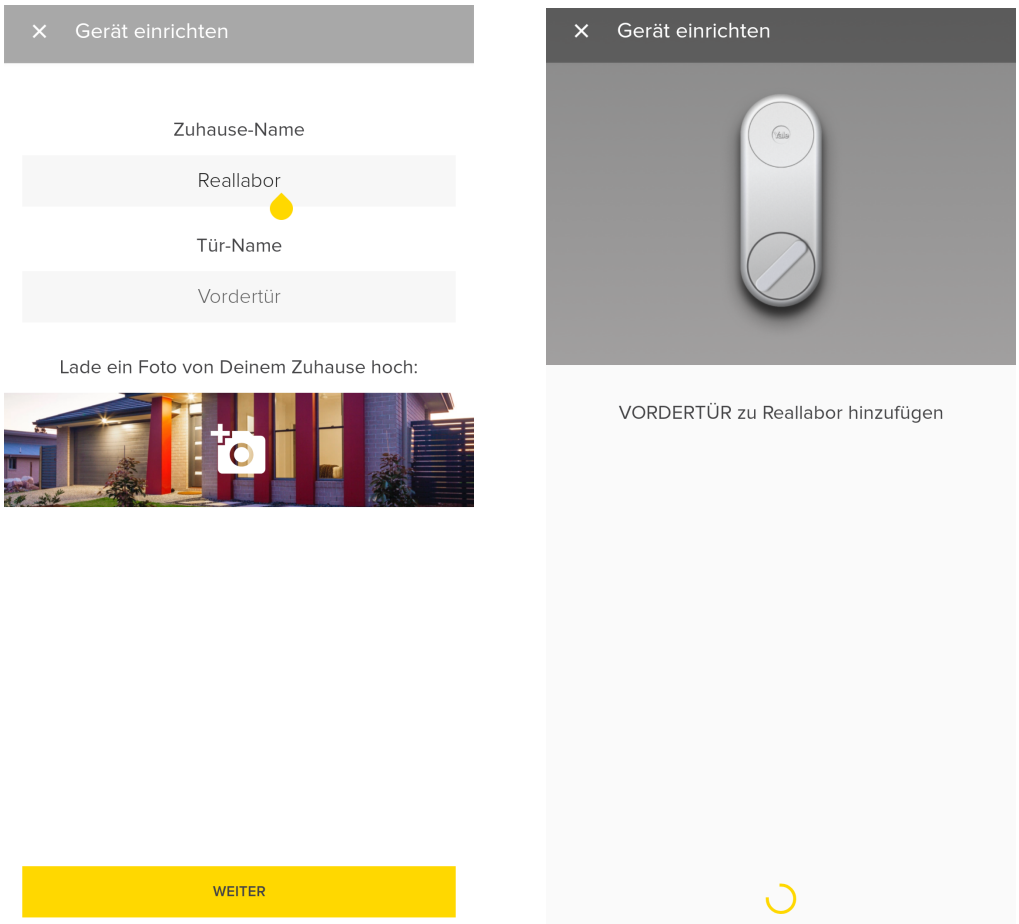



**Abbildung A.6:** Einbau des Yale Linus® Smart Schlosses [29]




**(a)** Vergeben von Namen für das Zuhause und die Tür

**(b)** Zugehörigkeit der Tür zum Zuhause

**Abbildung A.7:** Einrichtung des Geräts [29]

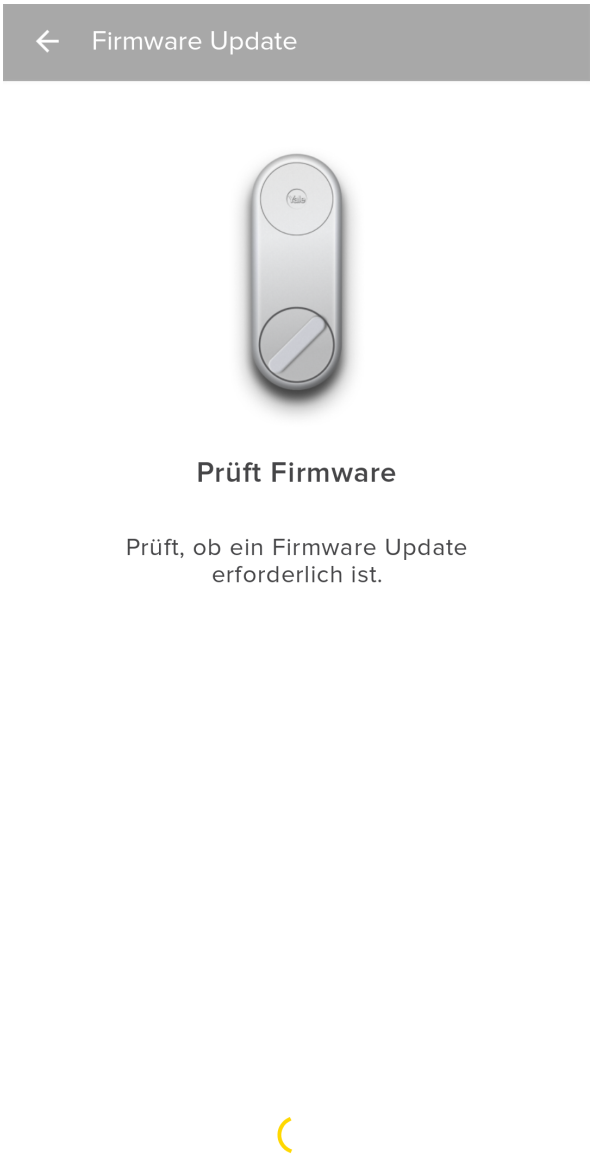
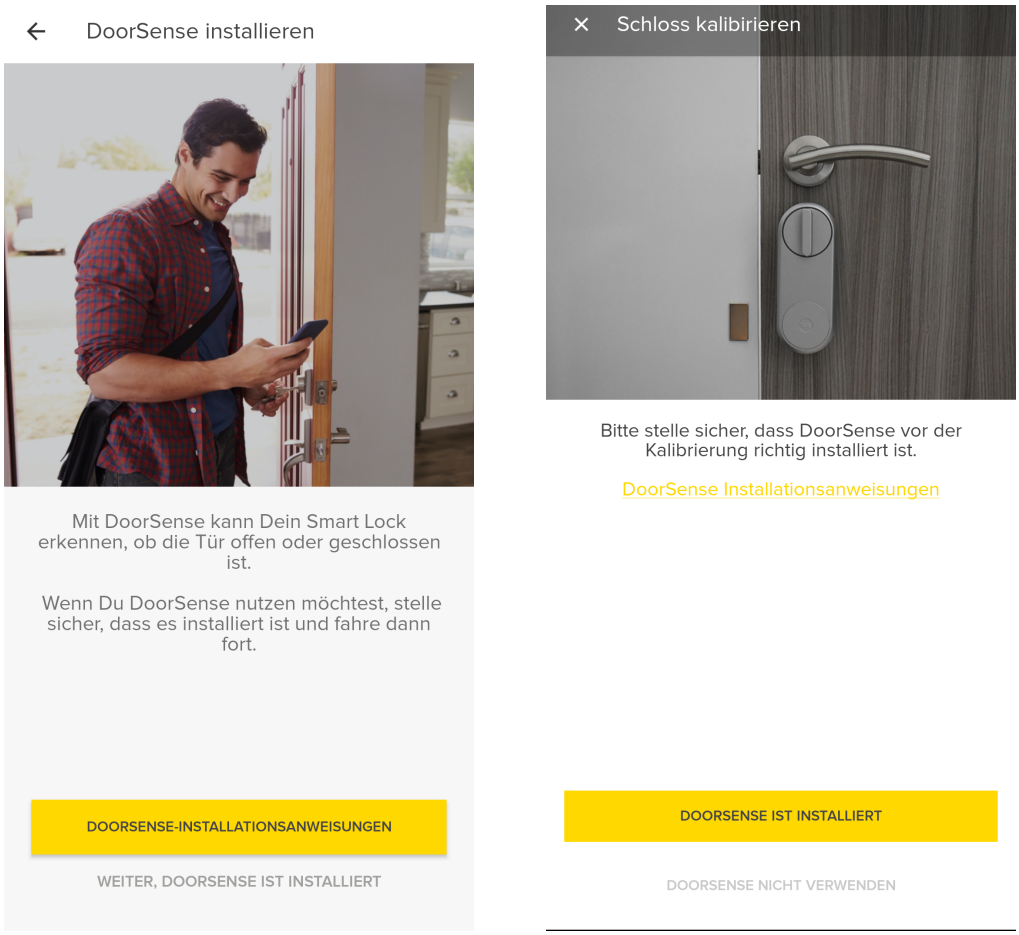



**Abbildung A.8:** Firmware-Update [29]




**(a)** DoorSense-Magnet anbringen und installieren

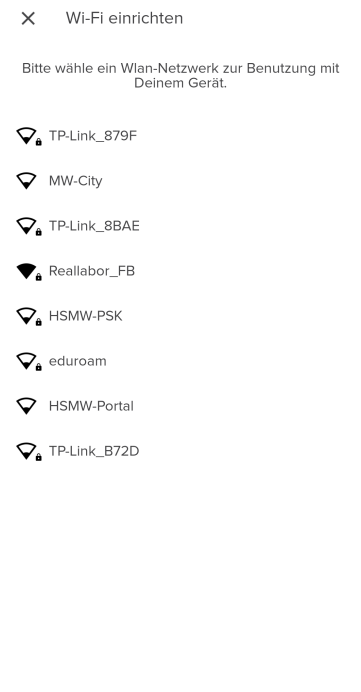
**(b)** Kalibrierung von DoorSense

**Abbildung A.9:** Einrichtung der DoorSense-Technologie [29]

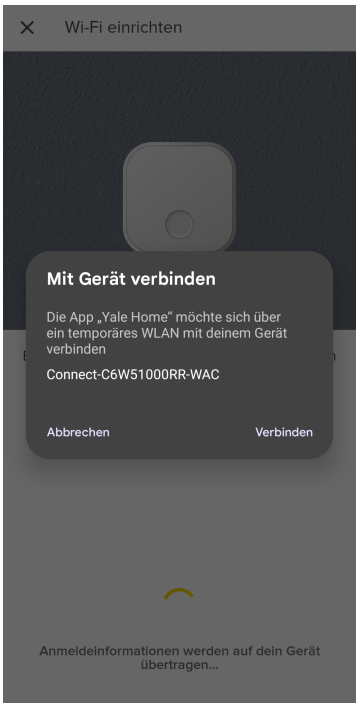

**(a)** An Strom anschließen


**(b)** Einschalten


**(c)** Integration in WLAN-Heimnetzwerk



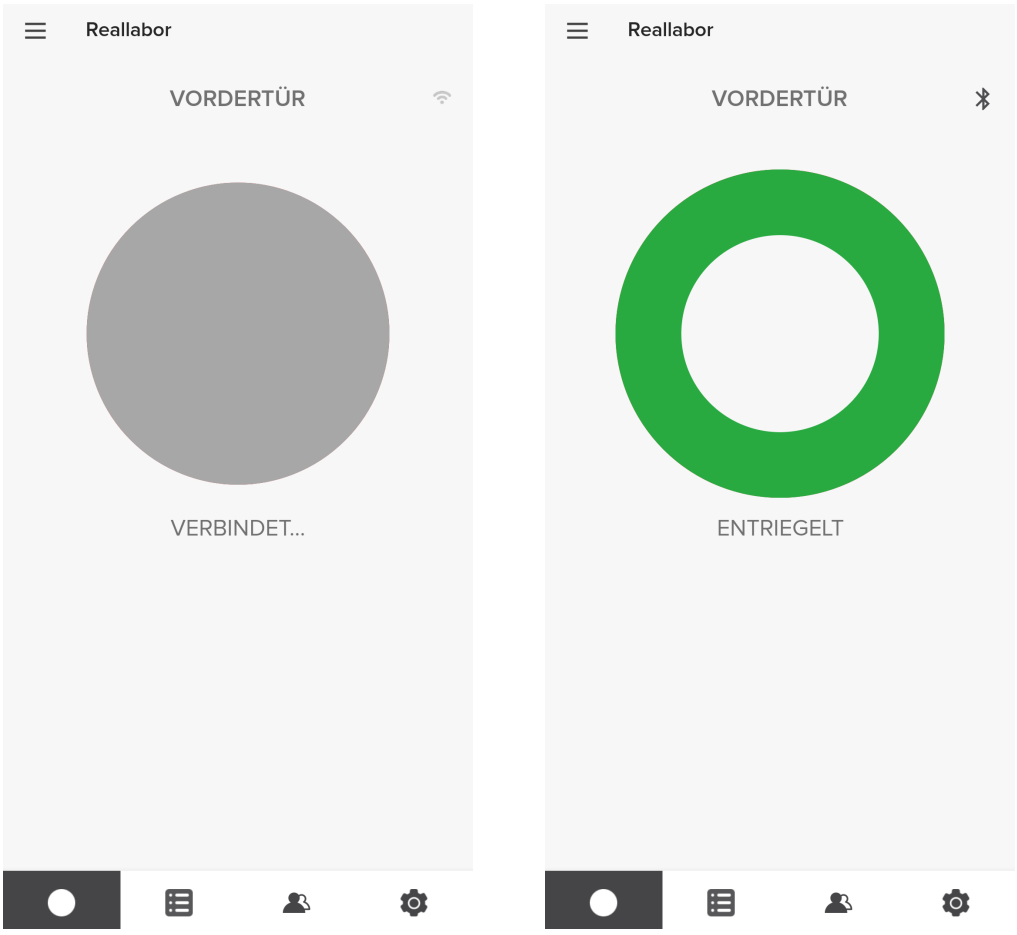

**(d)** WLAN-Netzwerk auswählen




**(e)** Verbinden der Yale Home-App mit temporären WLAN


**(f)** Signal-Test

**Abbildung A.10:** Einrichtung der Yale Connect Wi-Fi-Bridge [29]




**(a)** Verbindungsaufabu **(b)** Entriegelung

**Abbildung A.11:** Automatisches Testen des Schließvorgangs [29]

```
┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > Schliessen

┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > Öffnen

┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > Schließen

┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > 3x_Öffnen_Schließen

┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > Gastnutzer_Wiederkehrend

┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > Gastnutzer_Wiederkehrend_entziehen

┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > Gastnutzer_Temporär

┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > Gastnutzer_Temporär_entziehen

┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > Gastnutzer_Immer

┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > Gastnutzer_Immer_entziehen

┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > Gastnutzer_zu_Eigentümer

┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > Smart_Alert

┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > Gastnutzer_zu_Eigentümer_entziehen

┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > Auto_Lock_aktivieren

┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > Auto_Un_Lock_aktivieren

┌──(kali㉿kali)-[~]
└─$ adb shell "su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log" > Auto_Lock_10_sek
```

**Abbildung A.12:** ADB-Befehl zur Extraktion der Log-Dateien mit dazugehörigen Dateinamen

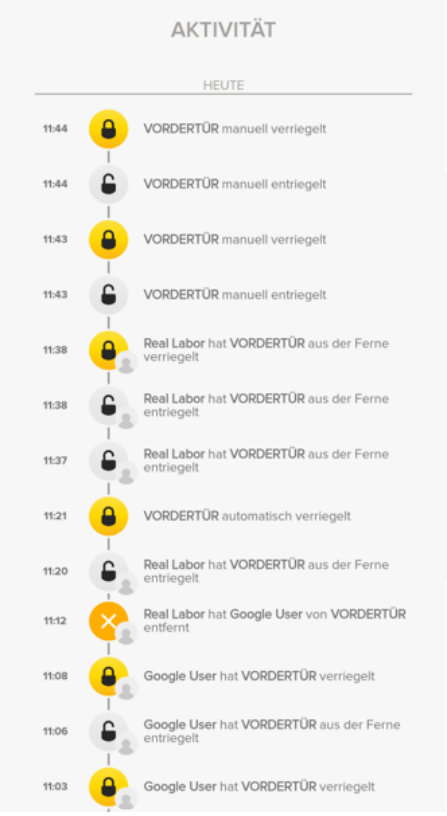
```
┌──(kali㉿kali)-[~]
└─$ adb shell pm list packages | grep yale
package:com.assaabloy.yale
```

**Abbildung A.13:** Grep-Befehl

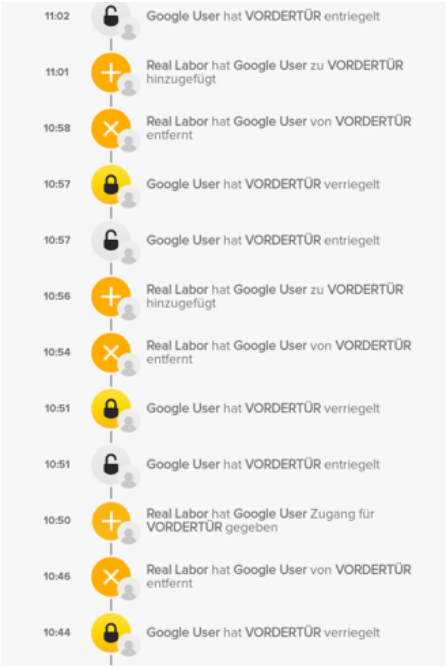
```
┌──(kali㉿kali)-[~]
└─$ adb shell
lynx:/ $ su
lynx:/ # cd /data/data/com.assaabloy.yale
lynx:/data/data/com.assaabloy.yale # ls
app_  app_textures  app_webview  cache  code_cache  databases  files  no_backup  shared_prefs
lynx:/data/data/com.assaabloy.yale # cp -r . /sdcard/backup-yale-linus
lynx:/data/data/com.assaabloy.yale # exit
lynx:/ $ exit

┌──(kali㉿kali)-[~]
└─$ adb pull /sdcard/backup-yale-linus
/sdcard/backup-yale-linus/: 166 files pulled, 0 skipped. 4.5 MB/s (16439617 bytes in 3.500s)
```

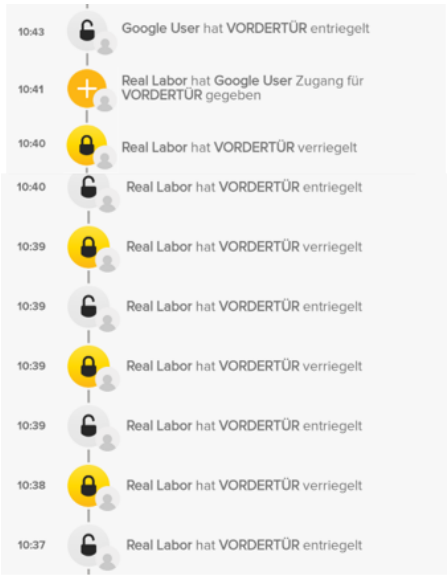
**Abbildung A.14:** Verwendete Befehle zur Extraktion der App-spezifischen Daten




**(a)** Teil 1




**(b)** Teil 2




**(c)** Teil 3

**Abbildung A.15:** Aktivitätsprotokoll der Yale Home App

# Anhang B: Tabellen

**Tabelle B.1:** Ereignisprotokoll der Cloud - Teil 1

| Time | Action | User | Phone Number | Email Address | Device Identification Number | Location |
|---|---|---|---|---|---|---|
| Mon Jul 28 2025 09:44:02 GMT+0000 (Coordinated Universal Time) | VORDERTÃœR manuell verriegelt | | | | LxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 09:44:01 GMT+0000 (Coordinated Universal Time) | VORDERTÃœR manuell entriegelt | | | | LxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 09:43:51 GMT+0000 (Coordinated Universal Time) | VORDERTÃœR manuell verriegelt | | | | LxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 09:43:50 GMT+0000 (Coordinated Universal Time) | VORDERTÃœR manuell entriegelt | | | | LxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 09:38:18 GMT+0000 (Coordinated Universal Time) | Real Labor hat VORDERTÃœR aus der Ferne verriegelt | Real Labor | 49xxxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 09:38:12 GMT+0000 (Coordinated Universal Time) | Real Labor hat VORDERTÃœR aus der Ferne entriegelt | Real Labor | 49xxxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 09:37:20 GMT+0000 (Coordinated Universal Time) | Real Labor hat VORDERTÃœR aus der Ferne entriegelt | Real Labor | 49xxxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 09:21:27 GMT+0000 (Coordinated Universal Time) | VORDERTÃœR automatisch verriegelt | | | | LxxxxxxxT | Europe/Berlin |

**Tabelle B.2:** Ereignisprotokoll der Cloud - Teil 2

| Time | Action | User | Phone Number | Email Address | Device Identification Number | Location |
|---|---|---|---|---|---|---|
| Mon Jul 28 2025 09:20:48 GMT+0000 (Coordinated Universal Time) | Real Labor hat VORDERTÃœR aus der Ferne entriegelt | Real Labor | 49xxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 09:12:50 GMT+0000 (Coordinated Universal Time) | Real Labor hat Google User von VORDERTÃœR entfernt | Real Labor | 49xxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 09:08:00 GMT+0000 (Coordinated Universal Time) | Google User hat VORDERTÃœR verriegelt | Google User | 49xxxxxxxxx0 | | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 09:06:41 GMT+0000 (Coordinated Universal Time) | Google User hat VORDERTÃœR aus der Ferne entriegelt | Google User | 49xxxxxxxxx0 | | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 09:03:30 GMT+0000 (Coordinated Universal Time) | Google User hat VORDERTÃœR verriegelt | Google User | 49xxxxxxxxx0 | | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 09:02:29 GMT+0000 (Coordinated Universal Time) | Google User hat VORDERTÃœR entriegelt | Google User | 49xxxxxxxxx0 | | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 09:01:03 GMT+0000 (Coordinated Universal Time) | Real Labor hat Google User zu VORDERTÃœR hinzugefÃ¼gt | Real Labor | 49xxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:58:55 GMT+0000 (Coordinated Universal Time) | Real Labor hat Google User von VORDERTÃœR entfernt | Real Labor | 49xxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:57:39 GMT+0000 (Coordinated Universal Time) | Google User hat VORDERTÃœR verriegelt | Google User | 49xxxxxxxxx0 | | LxxxxxxxxT | Europe/Berlin |

**Tabelle B.3:** Ereignisprotokoll der Cloud - Teil 3

| Time | Action | User | Phone Number | Email Address | Device Identification Number | Location |
|---|---|---|---|---|---|---|
| Mon Jul 28 2025 08:57:03 GMT+0000 (Coordinated Universal Time) | Google User hat VORDERTÃœR entriegelt | Google User | 49xxxxxxxxx0 | | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:56:16 GMT+0000 (Coordinated Universal Time) | Real Labor hat Google User zu VORDERTÃœR hinzugefÃ¼gt | Real Labor | 49xxxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:54:55 GMT+0000 (Coordinated Universal Time) | Real Labor hat Google User von VORDERTÃœR entfernt | Real Labor | 49xxxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:51:32 GMT+0000 (Coordinated Universal Time) | Google User hat VORDERTÃœR verriegelt | Google User | 49xxxxxxxxx0 | | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:51:10 GMT+0000 (Coordinated Universal Time) | Google User hat VORDERTÃœR entriegelt | Google User | 49xxxxxxxxx0 | | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:50:07 GMT+0000 (Coordinated Universal Time) | Real Labor hat Google User Zugang fÃ¼r VORDERTÃœR gegeben | Real Labor | 49xxxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:46:34 GMT+0000 (Coordinated Universal Time) | Real Labor hat Google User von VORDERTÃœR entfernt | Real Labor | 49xxxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:44:59 GMT+0000 (Coordinated Universal Time) | Google User hat VORDERTÃœR verriegelt | Google User | 49xxxxxxxxx0 | | LxxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:44:03 GMT+0000 (Coordinated Universal Time) | Google User hat VORDERTÃœR entriegelt | Google User | 49xxxxxxxxx0 | | LxxxxxxxxT | Europe/Berlin |

**Tabelle B.4:** Ereignisprotokoll der Cloud - Teil 4

| Time | Action | User | Phone Number | Email Address | Device Identification Number | Location |
|---|---|---|---|---|---|---|
| Mon Jul 28 2025 08:42:05 GMT+0000 (Coordinated Universal Time) | Real Labor hat Google User Zugang fÃ¼r VORDERTÃœR gegeben | Real Labor | 49xxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:40:05 GMT+0000 (Coordinated Universal Time) | Real Labor hat VORDERTÃœR verriegelt | Real Labor | 49xxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:40:00 GMT+0000 (Coordinated Universal Time) | Real Labor hat VORDERTÃœR entriegelt | Real Labor | 49xxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:39:55 GMT+0000 (Coordinated Universal Time) | Real Labor hat VORDERTÃœR verriegelt | Real Labor | 49xxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:39:51 GMT+0000 (Coordinated Universal Time) | Real Labor hat VORDERTÃœR entriegelt | Real Labor | 49xxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:39:46 GMT+0000 (Coordinated Universal Time) | Real Labor hat VORDERTÃœR verriegelt | Real Labor | 49xxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:39:42 GMT+0000 (Coordinated Universal Time) | Real Labor hat VORDERTÃœR entriegelt | Real Labor | 49xxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:38:37 GMT+0000 (Coordinated Universal Time) | Real Labor hat VORDERTÃœR verriegelt | Real Labor | 49xxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxT | Europe/Berlin |
| Mon Jul 28 2025 08:37:44 GMT+0000 (Coordinated Universal Time) | Real Labor hat VORDERTÃœR entriegelt | Real Labor | 49xxxxxxxxx3 | rxxxxxxxx.xxxxxxx@gmail.com | LxxxxxxxT | Europe/Berlin |

# Anhang C: Durchführungsprotokoll

## Protokollinformationen

- **Datum:** 28.07.2025
- **Uhrzeit:** 10:30 – 12:00 Uhr
- **Ort:** Mittweida, Labor
- **Versuchsleiter:** Emelie Hanske
- **Verwendete Geräte:**
  - Yale Linus® Smart Lock
  - Google Pixel 7a
  - Samsung Galaxy S22
  - DELL Laptop mit Linux Betriebssystem

## Versuchsverlauf

**Tabelle C.1:** Zeitlicher Ablauf von Bedienvorgängen am Yale Linus® Smart Lock

| Nr. | Aktion | Uhrzeit | Anmerkung |
|---|---|---|---|
| 1 | Schloss entriegeln | 10:37 | Öffnen.pcapng |
| 2 | Schloss verriegeln | 10:38 | Schließen.pcapng |
| 3 | Schloss 3x ent- und verriegeln | 10:39 - 10:40 | 3x_Öffnen_Schließen.pcapng |
| 4 | Zugangsberechtigung an Gastnutzer erteilen | 10:42 | Gastnutzer_Wiederkehrend.pcapng, Wiederkehrend: MO, 10:00 - 12:00 Uhr |
| 5 | Schloss entriegeln | 10:44 | Gastnutzer |
| 6 | Schloss verriegeln | 10:45 | Gastnutzer |
| 7 | Gastnutzer Zugangsberechtigung entziehen | 10:46 - 10:47 | Gastnutzer_Wiederkehrend_entziehen.pcapng |
| 8 | Zugangsberechtigung an Gastnutzer erteilen | 10:49 - 10:51 | Gastnutzer_Temporär, Temporär: 28.07.2025, 08:00 - 10:53 |
| 9 | Schloss entriegeln | 10:51 | Gastnutzer |
| 10 | Schloss verriegeln | 10:52 | Gastnutzer |
| 11 | Schloss entriegeln | 10:54 | Gastnutzer, außerhalb des angegebenen Zeitraums → Autorisierungsfehler erscheint |
| 12 | Gastnutzer Zugangsberechtigung entziehen | 10:55 | Gastnutzer_Temporär_entziehen.pcapng |
| 13 | Zugangsberechtigung an Gastnutzer erteilen | 10:57 | Gastnutzer_Immer.pcapng |
| 14 | Schloss entriegeln | 10:57 | Gastnutzer |
| 15 | Schloss verriegeln | 10:58 | Gastnutzer |
| 16 | Gastnutzer Zugangsberechtigung entziehen | 10:59 | Gastnutzer_Immer_entziehen.pcapng |
| 17 | Zugangsberechtigung an Gastnutzer erteilen | 11:00 - 11:02 | Gastnutzer_zu_Eigentümer.pcapng |

| | | | |
|---|---|---|---|
| 18 | Schloss entriegeln | 11:03 | Gastnutzer |
| 19 | Schloss verriegeln | 11:04 | Gastnutzer |
| 20 | Smart Alert erstellen | 11:05 - 11:06 | Smart_Alert.pcapng,<br>'Gerät wird von einem bestimmten Benutzer ent- oder verriegelt' |
| 21 | Schloss entriegeln | 11:07 | Gastnutzer,<br>Push-Up-Benachrichtigung auf dem Smartphone des Hauptnutzers |
| 22 | Schloss verriegeln | 11:08 | Gastnutzer |
| 23 | Gastnutzer Zugangsberechtigung entziehen | 11:13 | Gastnutzer_zu_Eigentümer_entziehen.pcapng |
| 24 | Schloss entriegeln | 11:14 | Gastnutzer,<br>Eigentlich kein Zugang mehr, funktioniert trotzdem |
| 25 | Schloss verriegeln | 11:14 | Gastnutzer,<br>Eigentlich kein Zugang mehr, funktioniert trotzdem |
| 26 | Schloss entriegeln | 11:15 | Autorisierungsfehler erscheint |
| 27 | Auto Lock aktivieren | 11:17 - 11:18 | Auto_Lock_aktivieren.pcapng,<br>auf 10 Sekunden eingestellt |
| 28 | Auto Unlock aktivieren | 11:19 - 11:20 | Auto_Unlock_aktivieren.pcapng |
| 29 | Schloss entriegeln,<br>Schloss verriegeln | 11:21 | Auto_Lock_10_sek.pcapng,<br>Schloss verriegeln über Auto Lock |
| 30 | Schloss aus Entfernung entriegeln | 11:37 | |
| 31 | Schloss aus Entfernung verriegeln | 11:38 | über Auto Lock |
| 32 | Schloss aus Entfernung entriegeln | 11:38 | |
| 33 | Schloss aus Entfernung verriegeln | 11:38 | |
| 34 | Schloss über Drehknauf manuell betätigt | 11:43 - 11:44 | |
| 35 | Ereignisprotokoll einsehen und aktualisieren | 11:47 | Ereignisprotokoll_anschauen.pcapng |

## Bemerkungen

Alle Aktionen wurden, sofern nicht anders in den Anmerkungen angegeben, vom Hauptnutzer durchgeführt.

# Literaturverzeichnis


[1] W. Babel, „Der Hype um die Welt der Automatisierung“, in *Industrie 4.0, China 2025, IoT*. Wiesbaden: Springer Vieweg Wiesbaden, 2021, S. 95–123, ISBN: 978-3-658-34717-8. DOI: 10.1007/978-3-658-34718-5. Adresse: https://doi.org/10.1007/978-3-658-34718-5.

[2] M. Barenkamp, J. H. Schoenke, N. Zarvic und O. Thomas, „IoT Best Practices“, in *IoT – Best Practices: Internet der Dinge, Geschäftsmodellinnovationen, IoT-Plattformen, IoT in Fertigung und Logistik*, S. Meinhardt und F. Wortmann, Hrsg. Wiesbaden: Springer Fachmedien Wiesbaden, 2021, S. 71–91, ISBN: 978-3-658-32439-1. DOI: 10.1007/978-3-658-32439-1_5. Adresse: https://doi.org/10.1007/978-3-658-32439-1_5.

[3] B. W. Wirtz, „Internet of Things“, in *Digital Business: Strategien, Geschäftsmodelle und Technologien*. Wiesbaden: Springer Fachmedien Wiesbaden, 2024, S. 261–299, ISBN: 978-3-658-41467-2. DOI: 10.1007/978-3-658-41467-2_7. Adresse: https://doi.org/10.1007/978-3-658-41467-2_7.

[4] Y. Kim, Y. Park und J. Choi, „A study on the adoption of IoT smart home service: using Value-based Adoption Model“, *Total Quality Management & Business Excellence*, Jg. 28, Nr. 9-10, S. 1149–1165, 2017. DOI: 10.1080/14783363.2017.1310708. Adresse: https://doi.org/10.1080/14783363.2017.1310708 (besucht am 05. 09. 2025).

[5] B. Schwerthalter und D. Helmold, „Cybersicherheitsmonitor 2024: Fokusthema Smarthome“, Bundesamt für Sicherheit in der Informationstechnik (BSI) und Polizeiliche Kriminalprävention der Länder und des Bundes (ProPK), Bonn, 2024. Adresse: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Digitalbarometer/CyMon-ProPK-BSI_2024_Kurzbericht_Smarthome.html?nn=1119026 (besucht am 02. 09. 2025).

[6] D. B. Rohleder, „Smart Home 2024“, Bitkom Research, Berlin, 2024. Adresse: https://www.bitkom.org/sites/main/files/2024-08/240822-Bitkom-Charts-Smart-Home-2024.pdf? (besucht am 05. 09. 2025).

[7] Sakshi, A. Malik und A. K. Sharma, „A survey on blockchain based IoT forensic evidence preservation: research trends and current challenges“, *Multimedia Tools and Applications*, Jg. 83, Nr. 14, S. 42 413–42 458, 2024, ISSN: 1573-7721. DOI: 10.1007/s11042-023-17104-z. Adresse: https://doi.org/10.1007/s11042-023-17104-z (besucht am 09. 09. 2025).

[8] W. Babel, „Internet of Things – IoT“, in *Systemintegration in Industrie 4.0 und IoT: Vom Ethernet bis hin zum Internet und OPC UA*. Wiesbaden: Springer Fachmedien Wiesbaden, 2024, S. 95–123, ISBN: 978-3-658-42987-4. DOI: 10.1007/978-3-658-42987-4_5. Adresse: https://doi.org/10.1007/978-3-658-42987-4_5.

[9] W. Babel, A. Hohorst, C. Jacob, S. Kukovev und M. Westermeier, *Technologie - Gestaltung - Umsetzung - Trends*. Haufe München, 2024, ISBN: 978-3-648-17674-0. DOI: 10.34157/978-3-648-17674-0. Adresse: https://doi.org/10.34157/978-3-648-17674-0.

[10] S. Alabdulsalam, K. Schaefer, T. Kechadi und N.-A. Le-Khac, „Internet of Things Forensics – Challenges and a Case Study“, in *Advances in Digital Forensics XIV*, Ser. IFIP Advances in Information and Communication Technology, Bd. 532, Springer International Publishing, 2018, S. 35–48. DOI: 10.1007/978-3-319-99277-8_3. Adresse: https://hal.inria.fr/hal-01988847.

[11] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis und E. K. Markakis, „A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues“, *IEEE Communications Surveys  Tutorials*, Jg. 22, Nr. 2, S. 1191–1221, 2020. DOI: 10.1109/COMST.2019.2962586.

[12] August Home, *August Wi-Fi Smart Lock*, https://august.com/products/august-wifi-smart-lock, 2025. (besucht am 24. 06. 2025).

[13] S. Hutchinson und U. Karabiyik, „Forensic Analysis of the August Smart Device Ecosystem“, in *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, 2020, S. 1–7. DOI: 10.1109/ISNCC49221.2020.9297346.

[14] J. Bays und U. Karabiyik, „Forensic Analysis of Third Party Location Applications in Android and iOS“, in *IEEE INFOCOM 2019 – Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Paris, France, 28. Juni 2019, S. 1–6. DOI: 10.48550/arXiv.1907.00074.

[15] A. Goudbeek, K.-K. R. Choo und N.-A. Le-Khac, „A forensic investigation framework for smart home environment“, in *Proceedings of the IEEE Conference*, IEEE, 2018, S. 1446–1451. DOI: 10.1109/TrustCom/BigDataSE.2018.00141.

[16] N. J. Apthorpe, D. Reisman und N. Feamster, „A smart home is no castle: Privacy vulnerabilities of encrypted IoT traffic“, in *Proceedings of the Workshop on Data and Algorithmic Transparency (DAT)*, Bd. abs/1705.06805, 2017. DOI: 10.48550/arXiv.1705.06805. arXiv: 1705.06805. Adresse: http://arxiv.org/abs/1705.06805.

[17] National Institute of Standards and Technology (NIST), „Mobile Device Data Population Setup Guide“, National Institute of Standards und Technology, Techn. Ber., 2019. Adresse: https://www.nist.gov/system/files/documents/2017/05/09/mobile%20device%20data%20population%20setup%20guide.pdf (besucht am 11. 06. 2025).

[18] Robert Bosch Smart Home GmbH. „Yale Linus® Smart Lock“, Bosch Smart Home. (2025), Adresse: https://www.bosch-smarthome.com/de/de/produkte/sicherheit/yale-linus-smart-lock/ (besucht am 27. 05. 2025).

[19] Yale Home, *Yale Linus Smart Door Lock*, https://www.yalehome.com/ae/en/products/smart-door-locks/linus-smart-door-lock/yale-linus-smart-door-lock, 2025. (besucht am 17. 06. 2025).

[20] Yale Home, *Yale Linus Smart Lock – Compatibility Checker*, 2025. Adresse: https://www.yalehome.com/hr/en/support/product-support/smart-locks/yale-linus/getting-started/linus-compatibility-checker (besucht am 03. 11. 2015).

[21] Cyberport GmbH. „Yale Linus Smart Lock Silber“, Cyberport. (2025), Adresse: https://www.cyberport.de/smart-home/sicherheit-alarm/smart-locks-zutrittssteuerung/yale/pdp/5c03-01h/yale-linus-smart-lock-silber.html? (besucht am 27. 05. 2025).

[22] T. Security. „Hardware vulnerabilities in smart locks: Teardown of the Yale Linus Smart Lock“, Tarlogic Security. (2022), Adresse: https://www.tarlogic.com/blog/hardware-vulnerabilities-smart-locks/ (besucht am 03. 05. 2025).

[23] Renesas Electronics Corporation. „DA14680 SmartBond Bluetooth Low Energy 4.2 SoC with Flash“, Renesas Electronics Corporation. (2025), Adresse: https://www.renesas.com/en/products/wireless-connectivity/bluetooth-low-energy/da14680-smartbond-bluetooth-low-energy-42-soc-flash?queryID=21652859df787ee4867b76640a92e5f0 (besucht am 30. 05. 2025).

[24] Yale Home, *Standort behandeln – Kompatibilitätsprüfer*, https://yalehome.de/kompatibilitatsprufer/standort-behandeln, 2025. (besucht am 02. 06. 2025).

[25] ASSA ABLOY Sicherheitstechnik GmbH, *Vor dem Kauf deines Linus Smart Lock*, https://www.youtube.com/watch?v=2KH7JjiGx_k, YouTube-Video, 20. Okt. 2020. (besucht am 03. 06. 2025).

[26] ASSA ABLOY Sicherheitstechnik GmbH. „Montage von Linus® Smart Lock an einem kompatiblen Zylinder“. (20. Okt. 2020), Adresse: https://www.youtube.com/watch?v=NS7uEmgprWs (besucht am 02. 06. 2025).

[27] ASSA ABLOY Sicherheitstechnik GmbH, *Montage Linus Smart Lock - Adjustable Zylinder*, https://www.youtube.com/watch?v=1rWASLM5NXc, YouTube-Video. (besucht am 03. 06. 2025).

[28] Yale Home, *Yale Home App Phone Compatibility*, 2025. Adresse: https://www.yalehome.com/global/en/trusted-innovation/yale-apps/yale-home-phone-compatibility (besucht am 05. 06. 2025).

[29] M. Teller, „Einrichten von Smart Home Geräten und einbinden dieser in OpenHAB als Vorarbeit für forensische Analysen von Datensicherheit“, Unveröffentlichter Bericht, erstellt im Rahmen des Pflichtpraktikums an der Hochschule Mittweida, 13. Dez. 2024.

[30] *Yale Smart Products – Datenschutzerklärung – Deutsch*. Adresse: https://www.yalehome.com/global/en/privacy-and-legal-center/user-terms-and-privacy-notice-for-yale-smart-products/europe-german/europe-yale-smart-products-privacy-notice-german (besucht am 09. 09. 2025).

[31] Yale. „FAQ zu Yale Linus® Smart Lock, What are different types of guest access?“, Yale Support. (2024), Adresse: https://support.yalehome.com/hc/en-us/articles/360050716792 (besucht am 18. 06. 2025).

[32] AA Ecosystem, *aaecosystem-account-ui: Login*, 2025. Adresse: https://account.aaecosystem.com/login (besucht am 31. 10. 2025).

[33] Yale Home, *Yale Access App Guide*, 2023. Adresse: https://www.yalehome.com/ae/download-center/yale-access-range/yale-access-app/Yale%20Access%20App%20Guide.pdf (besucht am 03. 11. 2025).

[34] Yale, *How to create, modify and delete smart alerts*, 2025. Adresse: https://www.yalehome.com/hr/en/support/product-support/smart-locks/yale-linus/everyday-use/how-to-create-modify-and-delete-smart-alerts? (besucht am 19. 06. 2025).

[35] K. K. Karmakar, V. Varadharajan, S. Nepal und U. Tupakula, „SDN Enabled Secure IoT Architecture“, in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, S. 581–585, ISBN: 978-3-903176-15-7.

[36] E. Schiller, A. Aidoo, J. Fuhrer, J. Stahl, M. Ziörjen und B. Stiller, „Landscape of IoT Security“, *Computer Science Review*, Jg. 44, S. 100 467, 2022. DOI: 10.1016/j.cosrev.2022.100467. Adresse: https://www.sciencedirect.com/science/article/pii/S1574013722000120 (besucht am 05. 05. 2025).

[37] B. V. S. Krishna und T. Gnanasekaran, „A systematic study of security issues in Internet-of-Things (IoT)“, in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2017, S. 107–111. DOI: 10.1109/I-SMAC.2017.8058318.

[38] R. Djouani, K. Djouani, F. Boutekkouk und R. Sahbi, „A Security Proposal for IoT integrated with SDN and Cloud“, in *2018 6th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 2018, S. 1–5. DOI: 10.1109/WINCOM.2018.8629727.

[39] A. Banks, E. Briggs, K. Borgendale und R. Gupta, Hrsg., *MQTT Version 5.0*, USA: OASIS Open, 7. März 2019. Adresse: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html (besucht am 11. 08. 2025).

[40] ASSA ABLOY / Yale, *Yale Linus® Smart Lock – Produktdatenblatt*, 2021. Adresse: https://www.assaabloy.com/ee/en/solutions/products/smart-locks/linus-smart-lock (besucht am 27. 06. 2025).

[41] S. Cheruvu, A. Kumar, N. Smith und D. M. Wheeler, „Connectivity Technologies for IoT“, in *Demystifying Internet of Things Security: Successful IoT Device/Edge and Platform Security Deployment*. Berkeley, CA: Apress, 2020, S. 347–411, ISBN: 978-1-4842-2896-8. DOI: 10.1007/978-1-4842-2896-8_5. Adresse: https://doi.org/10.1007/978-1-4842-2896-8_5.

[42] Yale Home. „Yale Connect Wi-Fi Bridge“. (2024), Adresse: https://www.yalehome.com/hr/en/products/smart-locks/yale-connect-wi-fi-bridge (besucht am 30. 06. 2025).

[43] S. S. Reddy Nalla, „Understanding Transport Layer Security (TLS): Safeguarding Secure Communication“, *Journal of Cybersecurity and Communication Systems*, Jg. 12, Nr. 1, S. 34–49, Apr. 2024. DOI: 10.1234/jccs.v12i1.2024.tls. Adresse: https://www.researchgate.net/publication/379538852_Understanding_Transport_Layer_Security_TLS_Safeguarding_Secure_Communication (besucht am 30. 06. 2025).

[44] B. Marques, „Smart Lock System for Door Access Control“, Magisterarb., Instituto Superior Técnico, University of Lisbon, 2022. Adresse: https://bernardocmarques.pt/assets/pdfs/work/tese/87634-bernardo-marques-dissertacao.pdf (besucht am 30. 06. 2025).

[45] E. Simeoni u. a., „A Secure and Scalable Smart Home Gateway to Bridge Technology Fragmentation“, *Sensors*, Jg. 21, Nr. 11, 2021, ISSN: 1424-8220. DOI: 10.3390/s21113587. Adresse: https://www.mdpi.com/1424-8220/21/11/3587 (besucht am 30. 06. 2025).

[46] J. Tosi, F. Taffoni, M. Santacatterina, R. Sannino und D. Formica, „Performance Evaluation of Bluetooth Low Energy: A Systematic Review“, *Sensors*, Jg. 17, Nr. 12, 13. Dez. 2017, ISSN: 1424-8220. DOI: 10.3390/s17122898. Adresse: https://www.mdpi.com/1424-8220/17/12/2898 (besucht am 01. 07. 2025).

[47] K. Townsend, C. Cufi, Akiba und R. Davidson, „Protocol Basics“, in *Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking*, B. Sawyer und M. Loukides, Hrsg., Sebastopol, CA, USA: O'Reilly Media, Inc., 2014, S. 15–34, ISBN: 978-1491949511.

[48] BluetoothSIG, „Vol 6: Core System Package [Low Energy Specification], Part A: Physical Layer Specification“, in *Specification of the Bluetooth® System, Covered Core Package Version: 5.0*, Kirkland, WA, USA: The Bluetooth Special Interest Group, 2016, S. 2532–2546. Adresse: https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/low-energy-controller/physical-layer-specification.html.

[49] K. Townsend, C. Cufi, C. Wang und R. Davidson, „Introduction“, in *Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking*, B. Sawyer und M. Loukides, Hrsg., Sebastopol, CA, USA: O'Reilly Media, Inc., 2014, S. 1–14, ISBN: 978-1491949511.

[50] K. Townsend, C. Cufi, C. Wang und R. Davidson, „GAP (Advertising and Connections)“, in *Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking*, B. Sawyer und M. Loukides, Hrsg., Sebastopol, CA, USA: O'Reilly Media, Inc., 2014, S. 35–50, ISBN: 978-1491949511.

[51] BluetoothSIG, „Vol 6: Core System Package [Low Energy Specification], Part B: Link Layer Specification“, in *Specification of the Bluetooth® System, Covered Core Package Version: 5.0*, Kirkland, WA, USA: The Bluetooth Special Interest Group, 2016, S. 2547–2694. Adresse: https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/low-energy-controller/link-layer-specification.html (besucht am 03. 11. 2025).

[52] Bluetooth SIG, *Vol 1: Architecture & Terminology Overview, Part A: Architecture, In Specification of the Bluetooth® System, Covered Core Package Version: 5.0*, The Bluetooth Special Interest Group, Kirkland, WA, USA, 2016, S. 161–264. Adresse: https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/architecture,-mixing,-and-conventions/architecture.html (besucht am 15. 07. 2025).

[53] Bluetooth Special Interest Group, *Core Specification 4.0*, 2010. Adresse: https://www.bluetooth.com/specifications/specs/core-specification-4-0/ (besucht am 16. 07. 2025).

[54] Bluetooth Special Interest Group, *Bluetooth Core Specification Version 4.1*, 2013. Adresse: https://www.bluetooth.com/specifications/specs/core-specification-4-1/ (besucht am 15. 07. 2025).

[55] Bluetooth Special Interest Group, *Bluetooth Core Specification Version 4.2*, 2014. Adresse: https://www.bluetooth.com/specifications/specs/core-specification-4-2/ (besucht am 15. 07. 2025).

[56] V. Plenk, „Protokolle: TCP/IP“, in *Angewandte Netzwerktechnik kompakt: Dateiformate, Übertragungsprotokolle und ihre Nutzung in Java-Applikationen*. Wiesbaden: Springer Fachmedien Wiesbaden, 2024, S. 157–183, ISBN: 978-3-658-43530-1. DOI: 10.1007/978-3-658-43530-1_9. Adresse: https://doi.org/10.1007/978-3-658-43530-1_9.

[57] J. Schwenk, „Transport Layer Security“, in *Sicherheit und Kryptographie im Internet: Theorie und Praxis*. Wiesbaden: Springer Fachmedien Wiesbaden, 2020, S. 191–227, ISBN: 978-3-658-29260-7. DOI: 10.1007/978-3-658-29260-7_10. Adresse: https://doi.org/10.1007/978-3-658-29260-7_10.

[58] J. Daemen und V. Rijmen, „The Design of Rijndael: The Advanced Encryption Standard (AES)“, in (Information Security and Cryptography), Second Edition, Information Security and Cryptography. Berlin, Germany: Springer, 2020, ISBN: 978-3-662-60768-8. DOI: 10.1007/978-3-662-60769-5. Adresse: https://link.springer.com/book/10.1007/978-3-662-60769-5.

[59] „Introduction to the AES“, in *Algebraic Aspects of the Advanced Encryption Standard*. Boston, MA: Springer US, 2006, S. 1–4, ISBN: 978-0-387-36842-9. DOI: 10.1007/978-0-387-36842-9_1. Adresse: https://doi.org/10.1007/978-0-387-36842-9_1.

[60] S. Qureshi, S. Tunio, F. Akhtar, A. Wajahat, A. Nazir und F. Ullah, „Network Forensics: A Comprehensive Review of Tools and Techniques“, *International Journal of Advanced Computer Science and Applications*, Jg. 12, Nr. 5, S. 879–887, 2021. DOI: 10.14569/IJACSA.2021.01205103. Adresse: http://dx.doi.org/10.14569/IJACSA.2021.01205103 (besucht am 26. 07. 2025).

[61] V. Ndatinya, Z. Xiao, V. R. Manepalli, K. Meng und Y. Xiao, „Network forensics analysis using Wireshark“, *International Journal of Security and Networks*, Jg. 10, Nr. 2, S. 91–106, 2015. DOI: 10.1504/IJSN.2015.070421. Adresse: https://doi.org/10.1504/IJSN.2015.070421 (besucht am 26. 07. 2025).

[62] R. Regupathy, „Android Debug Bridge (ADB)“, in *Unboxing Android USB: A Hands-On Approach with Real World Examples*. Berkeley, CA: Apress, 2014, S. 125–138, ISBN: 978-1-4302-6209-1. DOI: 10.1007/978-1-4302-6209-1_7. Adresse: https://doi.org/10.1007/978-1-4302-6209-1_7.

[63] O. Cinar, „Storing Data“, in *Android Quick APIs Reference*. Berkeley, CA: Apress, 2015, S. 171–198, ISBN: 978-1-4842-0523-5. DOI: 10.1007/978-1-4842-0523-5_7. Adresse: https://doi.org/10.1007/978-1-4842-0523-5_7.

[64] „Bluetooth Core Specification: Part B – Link Layer“, Bluetooth Special Interest Group, Techn. Ber., 2021. Adresse: https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-60/out/en/low-energy-controller/link-layer-specification.html? (besucht am 03. 09. 2025).

[65] *Bluetooth Assigned Numbers*, https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Assigned_Numbers/out/en/Assigned_Numbers.pdf, 2. Sep. 2025.

[66] IANA Time Zone Database, *zone1970.tab (tzdb 2025a)*, Version 2025a, Internet Assigned Numbers Authority (IANA), 2025. Adresse: https://data.iana.org/time-zones/tzdb-2025a/zone1970.tab (besucht am 18. 08. 2025).

[67] A. Barros, R. Almeida, T. Melo und M. Frade, „Forensic Analysis of the Bumble Dating App for Android“, *Forensic Sciences*, Jg. 2, Nr. 1, S. 201–221, 2022, ISSN: 2673-6756. DOI: 10.3390/forensicsci2010016. Adresse: https://www.mdpi.com/2673-6756/2/1/16.

[68] M. Ye, N. Jiang, H. Yang und Q. Yan, „Security Analysis of Internet-of-Things: A Case Study of August Smart Lock“, in *Proceedings of IEEE INFOCOM 2017 – IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Case study on August Smart Lock security vulnerabilities, University of Nebraska-Lincoln, USA, 2017, S. 499–504. DOI: 10.1109/INFCOMW.2017.8116427.

[69] A. Home. „Yale & August Smart Locks | Seamless Smart Home“. (2025), Adresse: https://august.com/pages/yale?srsltid=AfmBOoosvbc86f2ZwUD-1kdxxyUgIjfKEy4wigLVQ3-4xHicUTICZb_6 (besucht am 31. 10. 2025).

# Eidesstattliche Erklärung

Hiermit versichere ich – Emelie Charlotte Hanske – an Eides statt, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt oder anderweitig veröffentlicht.

| | |
|---|---|
| Mittweida, 09. November 2025 | |
| Ort, Datum | Emelie Charlotte Hanske, B.Sc. |