Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Fachbereich Informatik
Department of Computer Science

Masterthesis

Sicherheitsanalyse von Funkprotokollen in der Heimautomatisierung

von Kevin Fröhlich
(9019336)

25. Oktober 2018

Erstprüfer: Prof. Dr. Karl Jonas
Zweitprüfer: Prof. Markus Ullmann

Inhaltsverzeichnis

Abbildungsverzeichnis **IV**

Tabellenverzeichnis **IV**

Eidesstattliche Erklärung **V**

1 Einleitung **1**

2 Stand der Forschung **2**
2.1 Heimautomatisierung . . . 2
2.2 Aktuelle Funkprotokolle . . . 3
2.2.1 Bluetooth Low Energy . . . 4
2.2.2 ZigBee . . . 5
2.2.3 Weitere . . . 7
2.3 Schutzziele . . . 8
2.3.1 Vertraulichkeit . . . 8
2.3.2 Integrität . . . 10
2.3.3 Verfügbarkeit . . . 11
2.3.4 Authentizität . . . 11
2.3.5 Nicht-Vermehrbarkeit . . . 12

3 Bekannte Angriffe und Sicherheitslücken **13**
3.1 Discovery Angriff . . . 13
3.2 Blink Angriff . . . 14
3.3 Replay Angriff . . . 15
3.4 Brute Force und Fuzzing Angriff . . . 16
3.5 Energy Depletion Angriff . . . 17
3.6 Key Distribution Angriff . . . 19
3.6.1 Einprogrammierte Schlüssel . . . 20
3.6.2 Jamming Angriff . . . 21
3.6.3 Man-in-the-Middle Angriff . . . 21
3.6.4 Security Downgrade Angriff . . . 22
3.7 Reset Angriff . . . 23
3.8 Firmware-Update Angriff . . . 26
3.9 Decompile Angriff . . . 27
3.10 Sybil Angriff . . . 28
3.11 Weitere Angriffe . . . 29

4 Vorgehensweise und Sicherheitsanalyse **31**
4.1 Allgemeine Informationen . . . 31
4.2 Vertraulichkeit . . . 32
4.2.1 Initiierung der Verschlüsselung . . . 32
4.2.2 Verschlüsselung . . . 33
4.3 Integrität . . . 34
4.3.1 Veränderte Befehle . . . 34
4.3.2 Integritätsprüfungen . . . 35
4.4 Verfügbarkeit . . . 35
4.4.1 Blockieren des Funkkanals . . . 36
4.4.2 Nicht-Erreichbarkeit von Geräten . . . 36

4.5 Authentizität . . . 37
4.5.1 Benutzerauthentifizierung . . . 37
4.5.2 Authentizität durch Nähe zum Gerät . . . 37
4.5.3 Firmware-Updates . . . 38
4.6 Nicht-Vermehrbarkeit . . . 39
4.6.1 Wiedereinspielen von Befehlen . . . 39
4.6.2 Klonen/Fälschen von Geräten . . . 40

5 Vorbereitungen zur Sicherheitsanalyse 41
5.1 Auswahl Funkprotokoll . . . 41
5.2 Auswahl von Hard- und Software zur Analyse . . . 42
5.2.1 Hardware . . . 42
5.2.2 Software . . . 43
5.3 Anpassung der Analyse . . . 44

6 Anwendung der Sicherheitsanalyse 46
6.1 Magic Blue Bulb . . . 46
6.1.1 Sicherheitsanalyse . . . 47
6.1.2 Ergebnis . . . 49
6.2 Tapplock One . . . 50
6.2.1 Sicherheitsanalyse . . . 51
6.2.2 Ergebnis . . . 56
6.3 Eqiva Türschlossantrieb . . . 57
6.3.1 Sicherheitsanalyse . . . 58
6.3.2 Ergebnis . . . 63
6.4 Basetech Raumthermostat . . . 64
6.4.1 Sicherheitsanalyse . . . 65
6.4.2 Ergebnis . . . 68
6.5 eQ-3 Heizkörperthermostat . . . 69
6.5.1 Sicherheitsanalyse . . . 70
6.5.2 Ergebnis . . . 73
6.6 iHealth Blutdruckmessgerät . . . 74
6.6.1 Sicherheitsanalyse . . . 75
6.6.2 Ergebnis . . . 81
6.7 ABUS Fahrradschloss . . . 82
6.7.1 Sicherheitsanalyse . . . 83
6.7.2 Ergebnis . . . 84

7 Evaluation 86

8 Fazit 88

I Literaturverzeichnis 89

II Anhang 92
II.1 Befehle für Magic Blue Bulb . . . 92
II.2 Befehle für Tapplock One . . . 92
II.3 Befehle für Basetech Raumthermostat . . . 93
II.4 Befehle für eQ-3 Heizkörperthermostat . . . 93
II.5 Befehle für iHealth Blutdruckmessgerät . . . 94

Abbildungsverzeichnis

1 Datenpakete in Bluetooth Low Energy 5
2 Endpunkte in ZigBee 6
3 Analyse eines Bluetooth Low Energy Pakets in Wireshark 13
4 Replay Angriff bei Bluetooth Low Energy 16
5 Fuzzing Angriff bei Bluetooth Low Energy 17
6 Möglichkeit eines Replay Angriffs nach Neustart eines Geräts 18
7 Man-in-the-Middle Angriff auf Bluetooth Low Energy 22
8 Abstandsprüfung in Philips Hue 25
9 Reset Angriff bei Philips Hue 25
10 Sybil Angriff bei Bluetooth 29
11 Passwortbefehl bei BLE-Schloss 30
12 Texas Instruments CC2540 USB-Dongle 42
13 BBC micro:bit (MB80) 42
14 Texas Instruments SmartRF Packet Sniffer 43
15 Auflisten der Lese- und Schreibrechte mit BLEAH 44
16 Lampe Magic Blue Bulb 46
17 App der Magic Blue Bulb 46
18 Vorhängeschloss Tapplock One 51
19 App des Tapplock One 51
20 Vorgehensweise zum Öffnen des Schlosses im „Normalzustand“ 54
21 Eqiva Türschlossantrieb 57
22 App des Equiva Türschlossantriebs 57
23 Aufbau eines Befehls zum Aufschließen der Tür 60
24 Verbindungsaufbau zwischen App und Equiva Türschlossantrieb 63
25 Basetech Raumthermostat 65
26 App des Basetech Raumthermostats 65
27 eQ-3 Heizkörperthermostat 69
28 App des eQ-3 Heizkörperthermostats 69
29 iHealth Blutdruckmessgerät 75
30 App des iHealth Blutdruckmessgeräts 75
31 Vorgehensweise bei Datenabfrage neuer Messdaten 79
32 ABUS Fahrradschloss mit Fernbedienung 82

Tabellenverzeichnis

1 Zusammenfassung Analyse Magic Blue Bulb 50
2 Zusammenfassung Analyse Tapplock One 56
3 Zusammenfassung Analyse Eqiva Türschlossantrieb 63
4 Zusammenfassung Analyse Basetech Raumthermostat 68
5 Zusammenfassung Analyse eQ-3 Heizkörperthermostat 73
6 Zusammenfassung Analyse iHealth Blutdruckmessgerät 81
7 Zusammenfassung Analyse ABUS Fahrradschloss 85

Eidesstattliche Erklärung

Ich versichere an Eides statt, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Datum, Ort, Unterschrift

1 Einleitung

Durch die Heimautomatisierung (auch Smart Home genannt) können verschiedene Geräte in Haushalten miteinander vernetzt werden. Dadurch soll für den Benutzer vieles vereinfacht und automatisiert werden. Es wird auf diese Weise beispielsweise ermöglicht, Lampen in einer Wohnung, Türschlösser oder Steckdosen automatisch oder auch von einem mobilen Endgerät aus zu steuern. Die Heimautomatisierung ist ein sehr aktuelles Thema, da nach und nach immer mehr Geräte dieser Art auf den Markt gebracht werden. Geräte der Heimautomatisierung kommen auch immer mehr in Privathaushalten zum Einsatz, wie eine Studie von 2017 verdeutlicht [Schröder 2017]. Demnach gibt es bei Befragten zu diesem Thema auch Sorgen über die Sicherheit der Heimautomatisierung.

Verschiedene Angriffe auf solche Infrastrukturen hat es in den vergangenen Jahren bereits gegeben, beispielsweise einen Angriff mit einem Wurm, der Glühbirnen in einem Smart Home System befällt [Scherschel 2016] oder Angriffe auf smarte Schlösser, durch die Angreifer physikalischen Zugang zu geschützten Gegenständen, Räumen oder ganzen Gebäuden erhalten konnten [Rose Ramsey 2016]. Insgesamt wird also deutlich, wie relevant die Sicherheitsproblematik im Bereich der Heimautomatisierung sein kann - insbesondere auch in der näheren Zukunft, wenn die Verbreitung solcher smarten Geräte in Haushalten zunimmt.

Für die Kommunikation zwischen den Geräten - die oft kabellos stattfindet - werden unterschiedliche Funkprotokolle verwendet. Diese sorgen dafür, dass Befehle übertragen und ausgeführt werden können sowie dafür, dass Zustände z.B. von Sensoren an weitere Geräte oder eine zentrale Stelle übermittelt werden können. Je nach Protokoll gibt es verschiedene Sicherheitsmechanismen um die Kommunikation vor Angriffen zu schützen.

Das Ziel dieser Arbeit besteht darin, eine möglichst allgemeine Vorgehensweise für Funkprotokolle in der Heimautomatisierung zu entwerfen, mit der Sicherheitsprobleme sowohl in den jeweiligen Standards, als auch in der konkreten Implementierung bestimmter Geräte oder Hersteller entdeckt werden können. Dazu werden beschriebene Angriffe analysiert, die in der Vergangenheit bereits möglich waren. Mit diesen Informationen aus verschiedenen Protokollen der Heimautomatisierung soll eine allgemeine Vorgehensweise entwickelt werden, mit deren Hilfe es möglich sein soll Sicherheitslücken oder -probleme systematisch zu erkennen bzw. spezielle Implementierungen auf ihre Sicherheit zu untersuchen.

Für ein dann ausgewähltes Protokoll der Heimautomatisierung soll das Konzept angewandt werden und die Sicherheitsaspekte untersucht werden. Dafür erfolgt eine Auswahl der zu untersuchenden Geräte, aus denen eine Infrastruktur bestehen soll. Es werden also entsprechende Geräte in einer Smart Home Infrastruktur zusammengestellt, die das entsprechende Protokoll implementieren und verwenden. Mit diesen wird dann die Sicherheitsanalyse durchgeführt.

Zunächst einmal werden bekannte Funkprotokolle sowie relevante Schutzziele für die Heimautomatisierung in Kapitel 2 vorgestellt. In Kapitel 3 werden dann verschiedene Angriffe auf Infrastrukturen der Heimautomatisierung aus der Vergangenheit beschrieben und die Problematiken dieser erklärt. Aus diesen wird im darauf folgenden Kapitel 4 eine allgemeine Vorgehensweise erarbeitet, die dann - nach einigen getroffenen Vorbereitungen in Kapitel 5 - schließlich an verschiedenen Geräten in Kapitel 6 angewendet wird. Am Ende der Arbeit folgt in Kapitel 7 eine Evaluation der Vorgehensweise und Strategie sowie ein abschließendes Fazit in Kapitel 8.

2 Stand der Forschung

Zunächst einmal soll geklärt werden, wobei es sich bei der Heimautomatisierung handelt und welche Relevanz diese besitzt. Außerdem werden verschiedene Funkprotokolle, die typischerweise in diesem Bereich eingesetzt werden, vorgestellt und geeignete Schutzziele präsentiert, die in diesem Themenfeld optimalerweise versucht werden sollten zu erreichen.

2.1 Heimautomatisierung

Als Heimautomatisierung (oder auch Smart Home) wird die Vernetzung von verschiedenen Geräten in der häuslichen Umgebung und die dadurch entstehende Automatisierung von Wohngebäuden bezeichnet. Es sind viele Szenarien denkbar, in denen die Heimautomatisierung das Leben von Menschen vereinfacht.

Dabei können verschiedene Einsatzbereiche und Ziele verfolgt werden, die eine bessere Lebensqualität ermöglichen. Diese Bereiche sind nach [Bertko Weber 2017] hauptsächlich die Energieeffizienz, der Komfort, die Sicherheit und das Entertainment.

Zum einen kann die Energieeffizienz verbessert werden. Ein Beispiel dafür ist der Einsatz eines Smart Home Systems, um eine Heizung zu steuern. Wenn ohnehin niemand im Haus ist, wäre es z.B. denkbar, dass die Heizung ihre Leistung verringert und erst bevor jemand nach Hause kommt, die Temperatur im Haus wieder gesteigert wird oder Sensoren an den Fenster könnten melden, wenn Fenster geöffnet sind und so die Heizung automatisch herunterregeln. Dadurch kann Energie gespart werden.

Zum anderen kann auch der Komfort für alle Personen im Haus gesteigert werden, indem zum Beispiel Jalousien oder Rollladen automatisch bei Sonnenauf- oder -untergang oder aber bestimmten Uhrzeiten automatisch geöffnet oder geschlossen werden.

Nicht nur durch einzelne Geräte, sondern auch durch die Verknüpfung von mehreren Geräten und Sensoren ist es zudem auch möglich Sicherheitsfunktionen zu realisieren. Beispielsweise ist es möglich verschiedene Lampen im Smart Home so zu steuern, dass sie den Eindruck vermitteln es sei jemand zu Hause, obwohl dies gar nicht der Fall ist. So könnte dies als Präventivmaßnahme für Einbrüche dienen. Ebenso besteht die Möglichkeit Heimrauchmelder miteinander zu verknüpfen, sodass bei Auslösen eines Rauchmelders alle anderen ebenfalls ein akustisches Signal abgeben und alle Lichter im Gebäude angeschaltet werden.

Im Bereich des Entertainments könnten außerdem beispielsweise Lichter automatisch gedimmt werden, wenn bestimmte Geräte wie Fernseher eingeschaltet werden und Jalousien zu bestimmten Tageszeiten mit dem Einschalten des Fernsehers ebenfalls heruntergefahren werden.

Jedoch bietet die Heimautomatisierung auch viel Potenzial in der Zukunft. Für bestimmte Menschen kann sie auch eine Erleichterung der Alltagsaufgaben ermöglichen, die ohne diese Automatisierung vielleicht nur schwer zu bewältigen sind und so zu einer Steigerung der Lebensqualität führen. Bei Menschen mit Behinderung oder älteren Menschen könnte bereits das manuelle Öffnen oder Schließen von Jalousien eine Herausforderung darstellen. Hier wäre eine Sprachsteuerung z.B. eine Möglichkeit, mit der die betroffene Personengruppe die Jalousien auch ohne große körperliche Mühe öffnen oder schließen könnte.

Für die Heimautomatisierung gibt es verschiedene Systeme unterschiedlicher Anbieter. Die Funktionsweisen sind jedoch im Groben vergleichbar. Ein solches System baut meist auf einer eigenen Infrastruktur auf, die aus Sensoren und Aktoren besteht, die mit einer zentralen Einheit verbunden sind und ein eigenständiges Netzwerk bilden. Die Sensoren messen Werte in ihrer Umgebung oder Zustände und übermitteln diese an die Zentrale. Ein Beispiel für solche Sensoren ist ein Temperatursensor, der eine Umgebungstemperatur misst.

Aktoren sind Geräte, die bestimmte Aktionen ausführen können, z.B. eine Jalousie, die sich auf- oder zufahren lässt. Der Benutzer kann das System über die zentrale Einheit steuern und automatische Aktionen verwalten oder manuell Aktionen ausführen, beispielsweise über eine App auf dem Smartphone oder eine Weboberfläche. Die Zentrale stellt also eine Art Vermittler zwischen dem Benutzer und der Infrastruktur der Heimautomatisierung dar. Sie steuert dann die einzelnen Akteure und bezieht von Sensoren Werte, die ebenfalls Aktionen auslösen können.

Die Kommunikation zwischen den Geräten kann dabei entweder kabelgebunden oder kabellos stattfinden. Ein kabelgebundenes System erfordert das Verlegen eines Kabels, was häufig im Zuge von Sanierungsarbeiten oder Neubauten passiert. Dadurch entsteht eine gewisse Inflexibilität, falls später weitere Komponenten zu dem System hinzukommen oder Geräte an eine andere Stelle positioniert werden sollen. Entsprechende Kabelstränge müssten dann ebenfalls neu verlegt oder verändert werden, was viel Aufwand bedeuten kann. Die Kommunikation zwischen den Geräten wäre in diesem Fall jedoch besser von Außen geschützt, da sie nur mit einer Anbindung an das Kabel erfolgen kann.

Ein kabelloses System hingegen ermöglicht eine deutlich höhere Flexibilität und benötigt bis auf das einmalige Anbringen der Geräte keine weiteren Umbauten oder das Verlegen von Kabeln. Somit ist ein kabelloses System auch als Nachrüstungsmöglichkeit geeignet. Später hinzukommende Komponenten können relativ einfach hinzugefügt werden und die Positionsveränderung von Geräten ist ebenfalls ohne großen Aufwand möglich. Die Kommunikation erfolgt jedoch in diesem Fall über die Luft und kann daher auch von jedem Angreifer in der näheren Umgebung empfangen werden. Außerdem könnte es bei sehr hoher Nutzung bestimmter Funkfrequenzen (z.B. in Ballungsräumen) zu Störungen kommen.

Bei der Heimautomatisierung können Geräte auch ohne dauerhafte Stromeinspeisung betrieben werden, indem Akkus oder Batterien als Energiequellen eingesetzt werden. Somit können diese Geräte völlig unabhängig von ihrer Umgebung sein.

Auf dem Markt gibt es aktuell viele Systeme der Heimautomatisierung unterschiedlicher Anbieter. Dabei gibt es allerdings keine einheitlichen Standards. Die Kommunikation der kabellosen Systeme erfolgt je nach Auswahl des Anbieters mit verschiedenen Funkstandards, die nicht unbedingt miteinander kompatibel sind. Die bekanntesten und am häufigsten in der Heimautomatisierung verwendeten Funkstandards werden im nachfolgenden Kapitel vorgestellt.

Typische Geräte in der Heimautomatisierung sind Lampen, Schlösser (z.B Türschlösser), Thermostate und verschiedene Geräte mit verbautem ansteuerbarem Motor (z.B. Heizungsregler oder Rollladensteuerungen).

2.2 Aktuelle Funkprotokolle

Verschiedene Funkprotokolle finden je nach Anbieter Verwendung in Smart Home Systemen auf dem Markt. Durch die kabellose Kommunikation der Geräte miteinander gibt es bestimmte Anforderungen, die ein Funkprotokoll erfüllen sollte, damit es für den Einsatz in der Heimautomatisierung geeignet ist [Bertko Weber 2017].

Die Kommunikation sollte zuverlässig sein, d.h. die gesendeten Nachrichten sollten sicher beim Empfänger ankommen und nicht verloren gehen. Außerdem sollte das Protokoll einen möglichst niedrigen Energieverbrauch anstreben, sodass auch batteriebetriebene Geräte möglichst lange betrieben werden können, ohne dass ein Batteriewechsel vollzogen werden muss, der je nach Positionierung des Geräts möglicherweise einen größeren Aufwand bedeuten könnte. Auch die Sicherheit spielt eine große Rolle, da bei der Heimautomatisierung ebenfalls Geräte eingesetzt werden können, die wichtige und/oder sicherheitsrelevante Aufgaben

wahrnehmen. Es könnten z.B. Türschlösser angesteuert werden, die Zugang zu Wohngebäuden oder anderen zu schützenden Räumen ermöglichen könnten. Die Kommunikation sollte also so geschützt werden, dass sie von Angreifern nicht mitgelesen oder manipuliert werden kann und Angreifer nicht die Kontrolle über das System übernehmen können.

Es gibt aktuell keinen einheitlichen Funkstandard für alle Geräte in der Heimautomatisierung oder im Internet der Dinge allgemein. Jeder Hersteller verwendet unterschiedliche Protokolle für die drahtlose Kommunikation, teilweise sogar eigene Protokolle. Einige häufig in der Heimautomatisierung eingesetzten Funkprotokolle werden im Folgenden näher vorgestellt.

2.2.1 Bluetooth Low Energy

Eine in der kabellosen Datenübertragung häufig eingesetzte Technologie ist Bluetooth. Seit der Version 4.0 beinhaltet dieses auch eine Erweiterung Bluetooth Low Energy (BLE), die speziell Geräte mit möglichst sehr niedrigem Energiebedarf adressiert und nicht den Fokus - wie bei Bluetooth an sich (im Folgenden auch als klassisches Bluetooth bezeichnet) - auf eine möglichst hohe Übertragungsrate legt. [Heydon 2013] gibt im Folgenden eine Übersicht über die Funktionalitäten von BLE. Bluetooth Low Energy kann demnach auch als eigenständige Technologie betrachtet werden und eignet sich durch den niedrigen Energieverbrauch für den Einsatz in Geräten der Heimautomatisierung.

Dadurch, dass Geräte unterschiedliche Ressourcen besitzen, kann es Geräte geben, die entweder nur BLE oder nur das klassische Bluetooth unterstützen (Single Mode Devices) oder aber Geräte, die beides unterstützen (Dual Mode Devices).

Sowohl das klassische Bluetooth, als auch Bluetooth Low Energy arbeiten auf dem 2,4-GHz-Band, das ebenfalls von einigen anderen Technologien wie WLAN verwendet wird. Das 2,4-GHz-Frequenzband gehört weltweit zu den sogenannten ISM-Bändern - Frequenzbänder, die für Industrie, Wissenschaft und Medizin verwendet werden. Durch die größere Nutzung des Frequenzbereichs ist dieser anfälliger für Störungen, beispielsweise in viel genutzten Gegenden wie Innenstädten. Aus diesem Grund wird ein sogenanntes „Frequency Hopping“ durchgeführt, also ein schneller Wechsel in einer bestimmten festgelegten Reihenfolge über die möglichen Kanäle. Diese Reihenfolge tauschen beide Partner zu Beginn der Kommunikation aus, sodass beide den entsprechenden Datenkanälen in der richtigen Reihenfolge folgen können.

BLE besteht aus mehreren Schichten. Die erste und unterste Schicht bildet die physikalische Schicht (Physical Layer), auf der es Sender und Empfänger gibt. Das Frequenzband wird in 40 separate Kanäle mit Abständen von 2 MHz aufgeteilt. Darauf baut die nächste Schicht auf, die Verbindungsschicht (Link Layer). Dort gibt es vier Arten von Gerätetypen: Advertiser (Geräte, die Pakete senden), Scanner (Geräte, die Pakete empfangen), Slaves (Geräte, die mit einem Master verbunden sind, aber nur einfache Prozeduren durchführen können) und Master (Geräte, die komplexere Prozeduren durchführen). Es gibt drei Kanäle, die Geräte verwenden können, um eine Verbindung aufzubauen, falls sie noch nicht verbunden sind (Advertising-Kanäle) sowie 37 Kanäle für den Datenaustausch (Datenkanäle).

Die Daten werden zu Paketen zusammengefasst die folgendem Schema folgen: Präambel (zum Synchronisieren), Adressinformationen, Header, Daten und Prüfsumme (mit Cyclic Redundancy Check, CRC). Der Header enthält dabei Informationen wie z.B. eine Sequenznummer. Die Abbildung 1 stellt ein beispielhaftes Datenpaket in BLE grafisch dar.

Damit Endgeräte mit einem möglichst geringen Energiebedarf realisiert werden können, wird in BLE ein Server-Client-Konzept verfolgt. Dies erfolgt auf Ebene des Attribute Protocol Layers. Ein Server ist ein Gerät, das Daten (als Attribute/Characteristics, z.B. Temperaturwert) vorhält. Ein Client ist ein Gerät, das Daten von einem Server abfragen und verändern

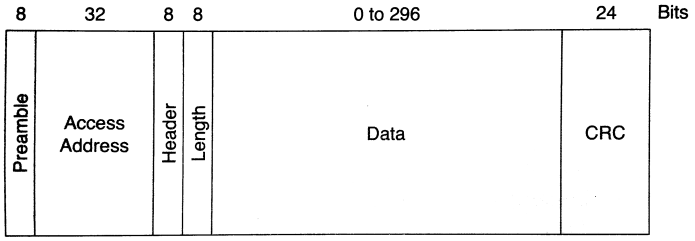

Abbildung 1: Datenpakete in Bluetooth Low Energy (aus [Heydon 2013])

kann. Je nach Status kann dieser nicht nur von außen gelesen sondern auch geschrieben werden (z.B. ein Sollwert für einen Temperaturregler kann gesetzt werden).

Die Besonderheit bei Bluetooth Low Energy ist es, dass Verbindungen nur vorübergehend sind. Wenn ein Client Daten benötigt, verbindet er sich kurzerhand mit einem Server, führt seine Aktionen aus und trennt die Verbindung wieder. Dadurch wird Energie gespart, da die Sende- und Empfangsfunktion des Servers außerhalb dieser Transaktionszeit nicht verwendet wird. Das bedeutet für die Heimautomatisierung mit Bluetooth Low Energy gibt es kein großes dauerhaftes Netzwerk, indem alle Geräte durchgehend miteinander kommunizieren, sondern der Benutzer stellt für den Zeitraum der Steuerung von Geräten eine Verbindung gezielt her (z.B. mit seinem mobilen Endgerät und einer entsprechenden App) und beendet diese danach wieder. Es handelt sich also um eine Verbindung zwischen zwei Geräten.

Das Attribut-Protokoll sorgt dafür, dass Zugriffe auf die Daten einheitlich erfolgen können. Dafür gibt es verschiedene Arten von Nachrichten, die möglich sind: Clients können Anfragen (mit resultierender Antwort) oder Befehle an Server senden und Server können Hinweise (mit resultierender Antwort) oder Benachrichtigungen an den Client senden. Es kann immer nur eine Transaktion (Anfrage mit Antwort) gleichzeitig ausgeführt werden, Befehle und Benachrichtigungen, die keine Antwort erwarten, können beliebig oft gesendet werden. Ein Attribut besteht aus einem Typ, einem Wert und einem Handle-Wert (einer eindeutigen Adresse). Dabei ist auch eine Rechtevergabe (Verteilung Zugriffs-, Authentifizierungs- und Autorisierungsberechtigungen) möglich. Es gibt Befehle, mit Hilfe deren der Client herausfinden kann was für Handle- und Typinformationen ein Server besitzt.

In der Regel gibt es bei der Verwendung von Bluetooth (Low Energy) keine zwischengeschaltete Zentrale als separates Gerät, da die meisten mobilen Endgeräte selbst die Möglichkeit besitzen Bluetooth und Bluetooth Low Energy Befehle zu senden und zu empfangen. Geräte der Heimautomatisierung können also direkt von mobilen Endgerät aus gesteuert werden. Dies setzt jedoch voraus, dass sich diese in entsprechender Reichweite befinden.

2.2.2 ZigBee

Ein sehr häufig verbreitetes Funkprotokoll in der Heimautomatisierung ist das Protokoll ZigBee. Es findet beispielsweise Verwendung in den bekannten Philips Hue Lampen und wird in [Krauße Konrad 2014] vorgestellt. Es baut demnach auf dem Funkstandard IEEE 802.15.4 auf, der die beiden untersten Schichten (Physikalische Schicht und MAC-Schicht) bereitstellt und setzt darauf zwei weitere Schichten (Netzwerkschicht und Anwendungsschicht) auf.

ZigBee arbeitet auf dem 2,4-GHz-Frequenzband (genauer 2400-2483,5 MHz), in Europa ist außerdem auch eine Frequenz von 868-868,6 MHz möglich. Ein ZigBee bzw. Netzwerk mit IEEE 802.15.4 wird auch als Personal Area Network (PAN) bezeichnet. Es hat eine eindeutige Identifikationsnummer (PAN-ID) mit einem Gerät als Koordinator. Dieser Koordinator verwaltet das Funknetz und ermöglicht beispielsweise den Beitritt neuer Geräte. Im Netzwerk gibt es zwei verschiedene Arten von Geräten: Geräte, die einen vollen Funktions-

umfang beinhalten (Full-Function Devices, FFD) und Geräte, die einen geringeren Funktionsumfang beinhalten (Reduced-Funktion Devices, RFD). Der zuvor genannte Koordinator ist ein Full-Function Device. Es kann im Netzwerk weitere FFDs geben, über die ein Beitritt ins Netzwerk ebenfalls möglich ist.

In IEEE 802.15.4 sind zwei Arten von Netzwerktopologien möglich: Eine Sterntopologie, d.h. der Datenaustausch findet immer über den Koordinator statt, und Peer-to-Peer, d.h. jedes Gerät kommuniziert mit jedem anderen Gerät im Umfeld (mit der Einschränkung, dass RFDs immer mit FFDs kommunizieren).

Die physikalische Schicht des IEEE 802.15.4 bildet die erste von vier Schichten in ZigBee. Sie ist für das Senden und Empfangen von Signalen verantwortlich und steuert die Umwandlung der digitalen Daten in Funksignale und umgekehrt. Die zweite Schicht ist die Media Access Control (MAC)-Schicht, die für eine zuverlässige Kommunikation im Netzwerk sorgt, indem z.B. die Adressierung geregelt wird und Prüfsummen gebildet werden. Jedes Gerät im Netzwerk besitzt eine eindeutige MAC-Adresse mit einer Länge von 64 Bit und sobald es dem Netzwerk beitritt bekommt es eine 16 Bit lange Kurzadresse, mit der die Kommunikation erfolgen kann, um die zu versendende Datenmenge zu reduzieren. Somit sind in einem ZigBee-Netzwerk maximal 65.536 Geräte möglich.

ZigBee spezifiziert drei unterschiedliche Rollen, die ein Gerät in einem Netzwerk annehmen kann: Koordinator, Router und Endgerät. Der Koordinator existiert in jedem Netzwerk genau einmal und ist zuständig für den Start des Netzwerks mit der Auswahl wichtiger Parameter (u.a. Auswahl Funkkanal, PAN-ID). Er muss ein FFD sein und übernimmt danach ebenfalls die Aufgaben eines Routers. Ein Router ist ebenfalls ein FFD und für die Weiterleitung von Datenpaketen verantwortlich. Somit ist er eine Art „Ansprechpartner“ für die Endgeräte. Über Router können zudem neue Geräte dem Netzwerk beitreten. Endgeräte können auch RFDs sein und kommunizieren nur über den Router, über den sie beigetreten sind. Sie können zwischenzeitlich in einem Schlafmodus verbleiben, um Energie zu sparen.

Auf der Anwendungsschicht gibt es sogenannte Endpunkte (bis zu 240 pro Gerät möglich), vergleichbar mit Netzwerkports, mit Hilfe deren Nachrichten bzw. Kommandos adressiert werden können. Beispielsweise könnte eine Fernbedienung einen Endpunkt 2 haben, mit dem eine Lampe geschaltet werden kann und bei einer Lampe gibt es einen Endpunkt 6, bei dem die Lampe sich an- oder ausschaltet. Zusätzlich könnte die Lampe an Endpunkt 5 auf einen Schalter reagieren. Die Abbildung 2 verdeutlicht dies. In einer sogenannten Binding-Tabelle wird gespeichert wie die Senderadresse und entsprechende Endpunkte mit Zieladresse und deren Endpunkten verknüpft sind.

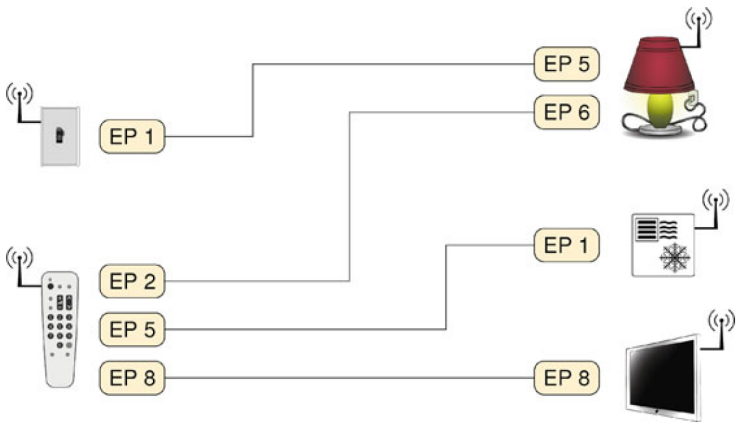

Abbildung 2: Endpunkte in ZigBee (aus [Krauße Konrad 2014, S.282])

Des Weiteren existieren in einer ZigBee Cluster Library sogenannte Cluster und Profile. Cluster definieren herstellerübergreifende Kommandos und Attribute auf Servern (also Geräten, die Daten vorhalten, die durch Clients gelesen und verändert werden können). So gibt es z.B. ein On/Off-Cluster, also ein Cluster mit zwei Zuständen, das zum Steuern von Lampen eingesetzt werden kann, die die Zustände an und aus haben. Damit sind also einheitliche Kommandos an die Endpunkte unabhängig vom Gerät oder Gerätehersteller möglich, indem Geräte Anfragen stellen können, welche anderen Geräte ein bestimmtes Cluster unterstützt. In Profilen werden weitere Eigenschaften für bestimmte Anwendungsbereiche wie spezielle Variablen, Konstanten oder Clusterkommandos festgelegt (z.B. Home Automation oder Light Link).

2.2.3 Weitere

Weitere bekannte Funkprotokolle in der Heimautomatisierung sind Z-Wave und EnOcean. Z-Wave ist ähnlich wie ZigBee ein Funkprotokoll für Netzwerke. Das bedeutet anders als bei Bluetooth (Low Energy) gibt es nicht nur eine kurzzeitige Verbindung zwischen zwei Geräten, sondern ein dauerhaftes Netzwerk mit mehreren Geräten. Dabei kann das Netzwerk als Mesh-Topologie aufgebaut sein, d.h. Geräte können untereinander miteinander kommunizieren und es gibt Routen, die zu bestimmten Geräten im Netzwerk verfolgt werden können [Yassein Mardini Khalil 2016].

Insgesamt sind 232 Geräte in einem Netzwerk mit Z-Wave möglich, wobei jedes davon eindeutig durch eine Home-ID (eindeutige Netzwerkidentifikation) in Kombination mit einer Node-ID (eindeutige Geräteidentifikation) bestimmt werden kann. Ebenfalls ähnlich zu ZigBee gibt es zwei Arten von Geräten: Controller und Slaves, wobei Controller Informationen über das Netzwerk, wie z.B. Routing-Tabellen verwalten und Slaves lediglich Befehle ausführen, die sie von Controllern erhalten [Yassein Mardini Khalil 2016].

Das Protokoll Z-Wave besteht aus den vier Ebenen Media Access Layer (Übermittlung der Daten auf Funkfrequenzebene mit Kollisionsvermeidung), Transfer Layer (Verbindungen zwischen Geräten, Single-/Multi-/Broadcasting), Routing Layer (Weiterleiten von Daten durch das Netzwerk) und Application Layer (Nutzdaten, Kodierung/Dekodierung, Befehle) [Yassein Mardini Khalil 2016].

Anders als ZigBee arbeitet Z-Wave jedoch nicht auf dem 2,4 GHz Frequenzband, sondern auf 868,4 MHz (in Europa und Asien) und ist ein proprietärer Standard des Unternehmens Zensys, das später von Sigma Designs übernommen wurde [Unwala Taqvi Lu 2018].

Ein weiterer Standard für die Kommunikation in der Heimautomatisierung bietet EnOcean. Er umfasst die Ebenen Physical Layer, Data Layer, Network Layer des OSI-Modells. Außerdem stellt die EnOcean Allianz sogenannte EnOcean Equipment Profile (EPP) auf Anwendungsebene bereit, die eine gute hersteller- und geräteübergreifende Kommunikation ermöglichen sollen. Alle Geräte kommunizieren miteinander in einer Mesh-Topologie. EnOcean erreicht eine Reichweite von etwa 300 Metern auf freier Fläche und 30 Metern in Gebäuden mit einer Übertragungsrate von 125 kB/s [Marksteiner et al. 2017].

Für die Kommunikation wird hauptsächlich das 868,3-MHz Frequenzband verwendet. Die Besonderheit bei EnOcean besteht darin, dass die EnOcean-Geräte meistens keine Stromversorgung per Kabel und auch nicht per Batterie oder Akku besitzen. Die Stromversorgung erfolgt durch sogenanntes „Energy Harvesting“ mittels Solarzellen, Windenergie oder kinetischer Energie [Bertko Weber 2017].

2.3 Schutzziele

Nach [Bedner Ackermann 2010] existiert eine Vielzahl von Schutzzielen, dessen Erreichung angestrebt werden kann. Die drei bekanntesten sind Vertraulichkeit, Integrität und Verfügbarkeit.

Auch in der Heimautomatisierung spielen Schutzziele eine große Rolle, denn es sollte versucht werden die Infrastruktur vor Angriffen zu schützen. Angreifer könnten beispielsweise ein Interesse daran haben bestimmte Geräte in der Heimautomatisierung zu manipulieren und selbst zu steuern. Ein Beispiel für einen solchen Angriff wäre der Angriff auf eine Türsteuerung, die mittels Smart Home gesteuert wird. Sollte es dem Angreifer gelingen, diese Türsteuerung zu manipulieren und die Tür so zu öffnen, könnte er Zugang zu einer Wohnung erhalten. Solche Angriffe hat es in der Vergangenheit bereits gegeben [Gierow 2015]. Es ist also notwendig, vor allem die Kommunikation in der Infrastruktur und die darin involvierten Geräten vor Angriffen zu schützen. Es sollte daher versucht werden Schutzziele zu definieren und diese zu erreichen.

Im Folgenden werden sinnvolle Schutzziele für die Heimautomatisierung charakterisiert und dargestellt welchen Sinn diese Schutzziele für die Heimautomatisierung haben sowie erläutert mit welchen Sicherheitsfunktionen diese üblicherweise erreicht werden können. Zudem werden die Sicherheitsfunktionen in den zuvor vorgestellten Funkprotokollen aufgezeigt.

2.3.1 Vertraulichkeit

Die Vertraulichkeit (engl. Confidentiality) definiert das Ziel, dass Informationen eines IT-Systems nur Befugten zugänglich sein sollen. Informationen können beispielsweise gespeicherte oder transportierte Nachrichteninhalte sein, die in diesem Schutzziel geschützt werden sollen [Bedner Ackermann 2010].

In der Heimautomatisierung werden Informationen zwischen einzelnen Geräten in der Infrastruktur ausgetauscht. Dabei kann es sich z.B. um Befehle handeln, die eine zentrale Einheit an einzelne Geräte sendet oder Messdaten, die ein Gerät an eine zentrale Einheit überträgt. Da die Kommunikation zumindest in den kabellosen Infrastrukturen von jedem Angreifer in der näheren Umgebung ebenfalls empfangen werden könnte, könnte dieser Informationen erlangen, die nicht für ihn bestimmt sind. Aus diesem Grund ist das Schutzziel der Vertraulichkeit auch in der Heimautomatisierung relevant.

Üblicherweise kann Vertraulichkeit durch Verschlüsselung erreicht werden. Diese kann entweder symmetrisch (selber Schlüssel zum ver- und entschlüsseln) oder asymmetrisch (unterschiedliche Schlüssel zum ver- und entschlüsseln) erfolgen. Bei der Verschlüsselung werden die zu schützenden Daten so verändert, dass es Angreifern ohne den Schlüssel nicht möglich sein soll die Daten sinnvoll zu interpretieren und so durch Mitlesen an Informationen zu kommen [Bedner Ackermann 2010].

Eine Übersicht über die Sicherheitsfunktionen in Bluetooth gibt ebenfalls [Heydon 2013]. In Bluetooth bzw. Bluetooth Low Energy kann zum Erreichen von Vertraulichkeit Verschlüsselung eingesetzt werden. Genauer werden dann die Nutzdaten (auch als Payload bezeichnet) in einer Verbindung mit dem Advanced Encryption Standard (AES) verschlüsselt. Nach [Heydon 2013, S.104ff] wird in Bluetooth Low Energy das Verfahren AES-128, also AES mit einem 128-Bit langen Schlüssel verwendet. Dabei wird der Cipher Block Chaining-Message Authentication Code Mode (CCM) genutzt (Counter with CBC-MAC), der dafür sorgt, dass jede Nachricht (auch bei gleichem Inhalt) unterschiedlich aussieht. Dies geschieht, indem ein einmalig gültiger Wert (Nonce) - bestehend aus einem Paket-Counter, einer Richtung und einem Initialisierungsvektor - sowie ein Block-Counter und ein fester Byte-Wert zusammengefügt (konkateniert) werden und so einen 16 Byte großen Block bilden. Dieser wird dann

verschlüsselt und per Exclusive-Or-Operation (XOR) mit den nächsten 16 Byte Nutzdaten verknüpft. Dies geschieht für alle Blöcke, bis alle Nutzdaten verarbeitet sind und so ergeben sich die verschlüsselten Nutzdaten, die bei jeder Verschlüsselung unterschiedlich aussehen, auch wenn die unverschlüsselten Nutzdaten die gleichen waren.

Der Einsatz von AES-128 (oder höher) wird vom Bundesamt für Sicherheit in der Informationstechnik (BSI) für neue Anwendungen empfohlen und der in BLE verwendete Cipher Block Chaining (CBC) Modus sowie Counter (CTR) Modus werden ebenfalls als geeignet beschrieben [BSI 2018, S.22ff]. Man könnte diese Verfahren also als aktuellen Stand der Technik bezeichnen.

Damit beide Geräte in einer Verbindung den selben Schlüssel besitzen und die Nachrichten ver- und entschlüsseln können, müssen die Schlüssel zuvor generiert und ausgetauscht werden, wie ebenfalls in [Heydon 2013] erklärt wird. Zunächst einmal findet ein Prozess statt, in dem beide Verbindungspartner Informationen austauschen, das sogenannte Pairing. Darin wird anfangs abgefragt welche Ein- und Ausgabemöglichkeiten (z.B. eine Anzeige von Ziffern auf einem Display) die beiden Geräte bieten, ob Out-Of-Band-Informationen (Informationen, die mit einer anderen Technologie als Bluetooth ausgetauscht werden können, z.B. Near-Field Communication) zur Verfügung stehen und welche Schlüssel angefragt werden.

Bei BLE gibt es mehrere Schlüssel, die je nach Status der Verbindung verwendet werden. Wird die Verbindung gerade zum ersten mal hergestellt (initialisiert), dann wird zunächst ein temporärer Schlüssel (Temporary Key, TK) berechnet. Je nach Kombinationen der Möglichkeiten der Geräte kann auf einem Gerät eine Ziffernfolge (bis zu 6-stellig) angezeigt werden, die der Benutzer auf dem anderen Gerät eingeben kann (Methode „Passkey Entry“), sodass das Mitschneiden der Schlüssel oder Man-In-The-Middle (MITM) Angriffe verhindert werden können. Diese Ziffernfolge bildet dann den TK. Wenn es jedoch keine Möglichkeit gibt, Ein- oder Ausgabefunktionalitäten zu verwenden, kann eine Methode „Just-Works“ verwendet werden, bei der der TK auf *0* gesetzt wird. Bei dieser Variante gibt es allerdings keinen Schutz vor MITM-Angriffen. Sind Out-Of-Band-Informationen ausgetauscht worden, können auch diese als TK verwendet werden.

Der temporäre Schlüssel, den nun beide Partner besitzen, wird verwendet um einen Kurzzeitschlüssel (Short-Term-Key, STK) zu berechnen. Genauer wird dieser Schlüssel verwendet, um zwei konkatenierte Zufallszahlen, den beide Kommunikationspartner ausgetauscht haben zu verschlüsseln. Dies bildet dann den sogenannten Short Term Key (STK).

Mit dem STK wird nun die Kommunikation beider Kommunikationspartner während des Pairing-Prozesses verschlüsselt, sodass die Partner verschlüsselt weitere Daten für die weitere Kommunikation austauschen können. Diese Daten sind ein Langzeitschlüssel (Long-Term-Key, LTK), der nach der initialen Kommunikation alle weiteren Nachrichten verschlüsselt, ein Schlüssel, der das Auflösen der Identität eines Geräts ermöglicht (Identity Resolving Key, IRK) sowie ein Schlüssel um Signaturen auflösen zu können (Connection Signature Resolving Key, CSRK). Diese Daten werden dann auf den Geräten gespeichert und können bei weiteren Verbindungen direkt verwendet werden, ohne dass die Schlüssel neu berechnet werden müssen. Bei der nächsten Verbindung der beiden Geräte wird dann die Verschlüsselung gestartet, nachdem ein Befehl „Start Encryption Request“ gesendet wird und der LTK wird für die Verschlüsselung aller Nachrichten verwendet.

In ZigBee wird ebenfalls Verschlüsselung der Nutzdaten in den Nachrichten mit AES-128 eingesetzt. Hierüber gibt [Krauße Konrad 2014] einen Überblick. Demnach gibt es zwei verschiedene Schlüssel, die den Kommunikationspartnern bekannt sein müssen. Sämtliche Kommunikation im Netzwerk wird mit einem Netzwerkschlüssel (Network Key) verschlüsselt. Zudem können Nachrichten zwischen einzelnen Teilnehmern mit einem Verbindungsschlüssel (Link Key) Ende-zu-Ende verschlüsselt werden.

Der Netzwerkschlüssel wird vom Trustcenter (einem Gerät im Netzwerk, das Sicherheits-

aufgaben wie die Schlüsselverteilung übernimmt, z.B. der Koordinator) beim Beitritt ins Netzwerk verteilt. Damit dieser Schlüsselaustausch nicht unverschlüsselt erfolgt (was jedoch auch möglich wäre), kann ein vorkonfigurierter Masterschlüssel, der fest in allen ZigBee-Geräten einprogrammiert ist, verwendet werden, um den Netzwerkschlüssel verschlüsselt zu übertragen.

Z-Wave verwendet für die Verschlüsselung der Nutzdaten auf Ebene der Anwendungsdaten (Application Layer) ebenfalls AES-128, jedoch ist die Verwendung von Verschlüsselung nicht zwingend und die Entscheidung den Herstellern überlassen. Der benötigte Schlüssel wird beim Eintreten in das Netzwerk durch das Verfahren Elliptic Curve Diffie-Hellman (ECDH) ausgetauscht [Unwala Taqvi Lu 2018].

Auch in EnOcean ist die Verschlüsselung der Nachrichten vorgesehen. Diese erfolgt ebenfalls mit AES mit einer Schlüssellänge von 128 Bit im Modus AES-CBC oder AES-CTR, wobei ein Initialisierungsvektor mit allen Bytes auf *0* gesetzt verwendet wird, was als unsicher gilt. Die Übertragung des dafür notwendigen Schlüssels geschieht entweder durch eine Übertragung im Klartext oder per pre-shared key, d.h. der Schlüssel ist beiden Kommunikationspartnern bereits bekannt [Marksteiner et al. 2017].

2.3.2 Integrität

Das Schutzziel Integrität (engl. Integrity) bezeichnet die Vollständigkeit und Korrektheit von Daten sowie die korrekte Funktionsweise des gesamten Systems. Das bedeutet, dass die Daten, die beim Empfänger ankommen, nicht unbemerkt verändert wurden und nicht beschädigt sind. Ein Angreifer soll diese also (zumindest nicht unbemerkt) manipulieren können [Bedner Ackermann 2010].

Auch dieses Schutzziel ist in der Heimautomatisierung relevant. Ein Benutzer könnte einen Befehl an ein Gerät senden lassen, mit dem eine Tür geschlossen wird. Dieser Befehl könnte jedoch von einem Angreifer so verändert werden, dass die Tür geöffnet wird und er so Zutritt zu einem Gebäude bekommt. Das Ziel sollte es also sein, dass Befehle oder andere Informationen von außen nicht unbemerkt verändert werden können, sodass hierdurch kein Schaden entstehen kann.

Integrität kann durch kryptografisch sichere Hashfunktionen realisiert werden. Diese bilden eine Abbildung der Daten auf einen Hashwert, der mit der Nachricht mitgesendet wird. Der Empfänger kann ebenfalls einen Hashwert aus den Daten der Nachricht bilden und diesen mit dem mitgesendeten Hashwert vergleichen [Bedner Ackermann 2010].

In Bluetooth Low Energy werden zur Realisierung des Schutzziels der Integrität zwei Maßnahmen ergriffen. Die erste Maßnahme soll vor Fehlern schützen. Dabei geht es allerdings um Fehler, die durch die Übertragung zu Stande kommen und nicht durch aktive Angreifer. Es wird eine Prüfsumme (Cyclic Redundancy Check, CRC) hinter jeder Nachricht hinzugefügt, mit der erkannt werden kann, wenn eine Nachricht fehlerhaft ist. Bei Angriffen ist dieser Schutz allerdings nicht ausreichend. Aus diesem Grund wird zusätzlich ein Message Integrity Check (MIC) an die verschlüsselten Nachrichten angehängt, der zum einen die Nachricht vor unbemerkter Manipulation schützt und zum anderen den Absender als legitim authentifiziert [Heydon 2013, S.107ff].

Der MIC wird ähnlich generiert wie die Nutzdaten der Nachricht, nur dass in diesem Fall nicht nur die Nutzdaten, sondern auch verschiedene andere Werte wie die Nonce, die Länge der Daten und Daten aus dem Header (der unverschlüsselt bleibt) als Blöcke miteinander verbunden und mit AES-CCM verschlüsselt werden. Daraus wird dann durch Verknüpfen dieser MIC-Blöcke per XOR mit dem letzten Block der Nutzdaten der Nachricht ein 32 Bit (4 Byte) großer MIC-Wert berechnet, der an die verschlüsselte Nachricht angehangen wird. Der legitime Empfänger ist in der Lage diese Berechnung ebenfalls durchzuführen, da er sowohl

im Besitz des gültigen Schlüssels ist, als auch die gesendeten Nutzdaten entschlüsseln konnte. Sofern diese manipuliert wurden, kommt ein anderer Wert für den MIC bei den Berechnungen heraus und das Paket wird verworfen sowie die Verbindung getrennt [Heydon 2013].

In ZigBee wird ebenfalls das AES-CCM-Verfahren verwendet, sodass die daraus generierbare Integritätsprüfung hier ebenfalls eingesetzt wird, um die Nachrichten vor unbemerkter Manipulation zu schützen. Je nach eingesetzter Sicherheitsstufe wird entweder gar keine Integritätsprüfung berechnet (Stufe 0) oder ein MIC-Wert mit einer Länge von 4, 8 oder 16 Byte berechnet [Krauße Konrad 2014]. In EnOcean wird für die Prüfung der Integrität AES-CMAC eingesetzt [Marksteiner et al. 2017].

2.3.3 Verfügbarkeit

Die Verfügbarkeit (engl. Availability) bezeichnet die ständige Verfügbarkeit des Systems und der Daten. Auf beides soll möglichst zeitnah von Autorisierten zugegriffen werden können [Bedner Ackermann 2010].

In der Infrastrukturen der Heimautomatisierung ist auch dieses Schutzziel von Relevanz, da ein Benutzer erwartet, dass seine Befehle an die Geräte möglichst zeitnah ausgeführt werden. Dabei kann es zudem auch um zeitkritische Aktivitäten gehen, wie beispielsweise das Auslösen eines Rauchmelders durch Rauch in der Wohnung. Ein solcher Rauchmelder könnte mit einer Smart Home Infrastruktur gekoppelt sein, die weitere Rauchmelder ebenfalls ein akustisches Signal ausgeben lässt, da sich einige Rauchmelder in Bereichen befinden, die die Bewohner nicht direkt einsehen können oder das Licht automatisch einschaltet. Eine Nicht-Verfügbarkeit des Systems oder eine zeitlich sehr verspätete Verarbeitung der Daten des Rauchmelders könnten bereits schwerwiegende Folgen haben.

Eine mögliche Maßnahme, um dieses Schutzziel zu erreichen, ist der Einsatz von Redundanzen, sodass Lasten, die auf Grund einer größeren Menge von Anfragen auf ein Gerät entstehen, besser verteilt werden können [Bedner Ackermann 2010].

Da ein Großteil der Geräte im Umfeld der Heimautomatisierung über keine ständige Stromversorgung verfügt, sondern mit Akkus oder Batterien betrieben werden, könnte es vorkommen, dass diese leer werden und das Gerät dadurch aus geht. Dies zu verhindern ist nur sehr schwer möglich, da eine Interaktion des Benutzers an dieser Stelle erforderlich ist, der z.B. Batterien austauscht oder Akkus wieder auflädt. Es ist jedoch möglich diesen durch geeignete Maßnahmen (leuchtende LEDs, Akustik, Anzeige im Display, Nachricht auf mobiles Endgerät per App, etc.) darauf hinzuweisen, wenn die Stromversorgung dem Ende entgegen geht.

Die Laufzeit von Akkus und Batterien kann durch ein Funkprotokoll, das möglichst wenig Energie verbraucht, begünstigt werden.

2.3.4 Authentizität

Als Authentizität (engl. Authenticity) wird die Sicherstellung bezeichnet, dass Daten wirklich von einem bestimmten angegebenen Absender stammen und somit die Identität, die der Absender angibt zu sein, korrekt ist [Bedner Ackermann 2010].

Im Fall der Heimautomatisierung sollte ein Gerät, das Befehle eines anderen Geräts entgegen nimmt und ausführt, sicher sein, dass es sich bei dem Absender des Befehls auch um eine autorisierte Stelle (z.B. die Zentrale) handelt und nicht um einen Angreifer.

Für gewöhnlich kann dieses Ziel dadurch erreicht werden, dass der Absender nachweisen kann, dass er z.B. im Besitz bestimmter Informationen oder Eigenschaften ist. Dies könnten eindeutige Benutzerkennungen mit Passwörtern sein [Bedner Ackermann 2010].

Um Authentizität zu erreichen wird in Bluetooth Low Energy der Schlüssel beim Einsatz

von Verschlüsselung initial ausgetauscht, sodass beide Partner im Besitz dieses Schlüssels sind. Bei weiteren Verbindungen der beiden ist der Besitz dieses Schlüssels, der sofort eine verschlüsselte Kommunikation ermöglicht also ein Beweis dafür, dass es sich bei dem Gegenüber um einen legitimierten Sender handelt. Beim Schlüsselaustausch wird Authentizität je nach Verfahren z.B. dadurch gewährleistet, dass ein Benutzer einen PIN-Code, der an einem Gerät generiert wird, am anderen Gerät eingibt [Heydon 2013, S.241f].

Die Authentizität erfolgt in ZigBee wie in BLE durch den Besitz eines Schlüssels - in diesem Fall Besitz des gültigen Netzwerkschlüssels und des Verbindungsschlüssels -, mit dem sowohl die Nachrichteninhalte entschlüsselt werden können, als auch der MIC-Wert überprüft und selbst generiert werden kann [Krauße Konrad 2014].

In Z-Wave gibt es drei verschiedene Sicherheitsklassen, zu denen Geräte je nach Authentifizierung zugeteilt werden: „S2 Access Control“, „S2 Authenticated“ und „S2 Unauthenticated“. Jede dieser Klassen besitzt einen eigenen Netzwerkschlüssel, sodass Geräte nur mit anderen Geräten der selben Klasse kommunizieren können. Ein Gerät kann auch mehreren Klassen angehören [Unwala Taqvi Lu 2018].

Bei EnOcean wird Authentifikation eines legitimen Benutzers durch den Einsatz des AES-CMAC-Verfahrens erreicht, bei dem der Benutzer den Schlüssel zum Ver- und Entschlüsseln der Kommunikation kennen muss [Marksteiner et al. 2017].

2.3.5 Nicht-Vermehrbarkeit

Das Schutzziel der Nicht-Vermehrbarkeit (engl. Non-Propagation) bedeutet, dass Informationen von Angreifern nicht kopiert und zu einem späteren Zeitpunkt erneut eingespielt werden können. In sogenannten Replay-Angriffen könnten Angreifer Nachrichten aufzeichnen und diese später erneut einspielen, sodass diese ein weiteres mal ausgeführt werden, ohne dass ein autorisierter Benutzer diese abgesendet hat [Bedner Ackermann 2010].

Auch dieses Schutzziel findet Relevanz in der Heimautomatisierung. Ein Beispiel wäre die Steuerung einer Tür innerhalb einer Heimautomatisierung. Ein Angreifer könnte den Befehl zum Öffnen dieser Tür aufnehmen, zu einem späteren Zeitpunkt erneut einspielen und sich so Zugang zu einer Wohnung verschaffen.

Um den Einsatz von Replay-Angriffen und im Allgemeinen die Vermehrbarkeit von Informationen zu verhindern, können große und nicht vorhersagbare einmalig gültige Zeichenketten - sogenannte Nonces - verwendet werden. Diese sind nach dem einmaligen Ausführen nicht mehr gültig und ein weiterer Befehl, der eine ungültige Nonce verwendet, würde verworfen und nicht ausgeführt [Bedner Ackermann 2010].

Andere Beispiele für Maßnahmen gegen Replay-Angriffe sind z.B. Counter oder Zeitstempel, die mit jeder Nachricht hochgezählt bzw. an die aktuelle Zeit angepasst werden, um so zu verhindern, dass alte Nachrichten eingespielt werden. Counter und Nonces werden beispielsweise in Bluetooth Low Energy im AES-CCM-Verfahren eingesetzt [Heydon 2013, S.104ff].

Auch in ZigBee wird zur Nicht-Vermehrbarkeit das Verfahren AES-CCM eingesetzt, das auch bei gleichem Klartextinhalt andere verschlüsselte Nachrichten erzeugt. Somit sehen Nachrichten immer anders aus. Aufgenommene Nachrichten variieren durch den Einsatz eines Frame-Counters und den Einsatz von AES-CCM und bei niedrigerem Frame-Counter als der in der zuletzt erfolgreich entgegengenommenen Nachricht wird die Nachricht verworfen. Ein aufnehmen und späteres erfolgreiches Einspielen von aufgenommenen Nachrichten wird dadurch also verhindert [Krauße Konrad 2014].

Durch den Einsatz eines Counters in den eingesetzten AES-Verfahren (CMAC) zur Verschlüsselung ändern sich Nachrichteninhalte in EnOcean auch bei gleichen Inhalten, wodurch diese gegen Wiedereinspielen alter Befehle geschützt werden [Marksteiner et al. 2017].

3 Bekannte Angriffe und Sicherheitslücken

In der Vergangenheit wurden bereits verschiedene Angriffe auf Infrastrukturen der Heimautomatisierung durchgeführt und Sicherheitslücken aufgedeckt. Diese sind in der Literatur beschrieben. In [Ray et al. 2018] wurde beispielsweise aus verschiedenen Angriffen in Bluetooth Low Energy ein Framework entwickelt, mit dem BLE-Geräte im Industrieumfeld auf Schwachstellen getestet werden können. Die relevantesten Angriffe - insbesondere mit Blick auf die Funkprotokolle in der Heimautomatisierung - sind im Folgenden beschrieben. Aus der Analyse der vergangenen Angriffe wird im nächsten Kapitel eine strukturierte Vorgehensweise erarbeitet, mit der Systeme der Heimautomatisierung analysiert werden können.

3.1 Discovery Angriff

Die Verwendung von Luft als Übertragungsmedium für die Kommunikation in der Heimautomatisierung an Stelle von Kabeln ermöglicht es Angreifern im näheren Umfeld sämtliche Kommunikation ebenfalls zu empfangen. Je nach Funkprotokoll und Einstellungen bzw. verwendeten Sicherheitsfunktionen im Netzwerk kann ein Angreifer so Informationen über das Netzwerk und die darin befindlichen Geräte erhalten. Er könnte also passiv alle Nachrichten aufnehmen und analysieren, um beispielsweise Adressen von Geräten zu erhalten.

Falls im Funkprotokoll Verschlüsselung eingesetzt wird, können die Nachrichten zwar ebenfalls empfangen werden, der eigentliche Inhalt bleibt ohne den richtigen Schlüssel jedoch unverständlich. Da allerdings in der Regel nur die Nutzdaten (Payload) verschlüsselt sind, könnten andere Nachrichtenteile wie Quell- und Zieladressen dennoch analysiert werden. 2014 war Verschlüsselung in ZigBee-Netzwerken noch nicht so sehr verbreitet, sodass hier auch die Nutzdaten mitgelesen und analysiert werden konnten [Olawumi et al. 2014].

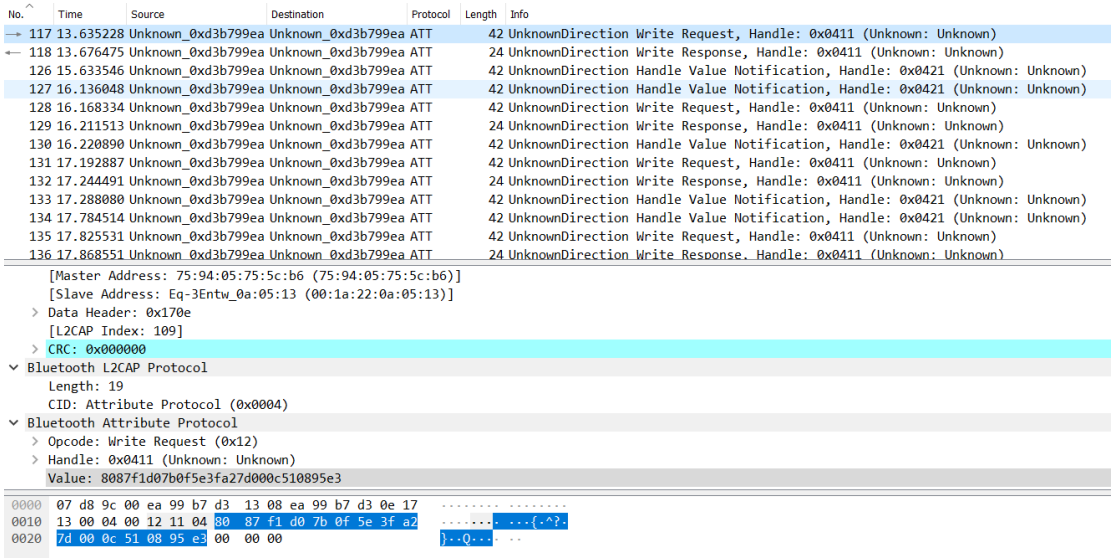

Abbildung 3: Analyse eines Bluetooth Low Energy Pakets in Wireshark

Ein Beispiel für die Analyse eines aufgenommenen Pakets mit der Software Wireshark ist in Abbildung 3 dargestellt. In diesem Fall handelt es sich um Pakete des Protokolls Bluetooth Low Energy.

Zudem könnte ein Angreifer beispielsweise bei ZigBee-Netzwerken auch aktiv tätig werden, indem er sogenannte Beacon Requests aussendet. Solche Befehle sind dafür vorgesehen, dass

Geräte, die einem neuen Netzwerk beitreten möchten, Informationen über dieses erhalten. So kann der Benutzer ein bestimmtes Netzwerk auswählen und beitreten. Mit Aussenden eines solchen Beacon Requests sendet der Koordinator des Netzwerks Informationen wie PAN-ID, Adressen von Koordinator und Routern, Stack Profile sowie Version als Antwort zurück [Olawumi et al. 2014].

Somit kann ein Angreifer allein durch Abhören der Kommunikation und ggf. durch Empfangen von Beacon Requests Informationen über das Netzwerk und die darin befindlichen Geräte erhalten. Es wird lediglich Hardware benötigt, die auf der Frequenz des Protokolls die Nachrichten empfängt und Software, die diese übersichtlich aufbereitet.

Außerdem kann es mit reinem Mithören bereits möglich sein sensible Informationen zu erhalten. Nämlich dann, wenn Geräte unverschlüsselte Daten wie Passwörter senden. Der Vortrag von [Rose Ramsey 2016] stellt verschiedene smarte Schlösser z.B. für Türen vor, die mit Bluetooth Low Energy gesteuert werden können. Nur sehr wenige Schlösser haben demnach angemessene Sicherheitsvorkehrungen getroffen, um vor Angriffen zu schützen. Bei vier Schlössern wird das Schloss mit einem Passwort geschützt, das der Benutzer in seiner App eingeben muss, um das Schloss zu öffnen. Dieses Passwort wird jedoch unverschlüsselt im Klartext per Write-Befehl auf das Schloss übertragen. Somit kann das Passwort von jedem Angreifer in der Nähe mitgelesen werden.

3.2 Blink Angriff

In der Arbeit von [Morgner Mattejat Benenson 2017] geht es um verschiedene Angriffe, die auf mit ZigBee gesteuerte Lichtsysteme anwendbar sind. Im Speziellen gibt es bei ZigBee Befehle, die zwischen Geräten, die sich in verschiedenen Netzwerken befinden, ausgetauscht werden können: sogenannte Inter-PAN Frames. Diese sorgen beispielsweise für die Möglichkeit, dass ein Gerät, das sich nicht in einem Netzwerk befindet, mit diesem kommunizieren kann, um anschließend in das Netzwerk beizutreten. Da das Gerät, das sich noch nicht im neuen Netzwerk befindet, noch nicht den Netzwerkschlüssel besitzen kann, findet diese Kommunikation unverschlüsselt und unauthentifiziert statt.

Um ein Gerät für den Benutzer leichter identifizierbar zu machen, wenn er es beispielsweise in sein Netzwerk hinzufügen möchte, existiert bei Geräten, die ZigBee Light Link (ZLL) verwenden, die Möglichkeit sich selbst zu identifizieren. Dieses Identifizieren kann mit dem Aussenden des Befehls „Identify Request“ eingeleitet werden. Da es sich bei den Geräten, die ZLL verwenden, um Lampen handelt, können diese sich gegenüber dem Benutzer sehr einfach identifizieren, indem sie anfangen zu blinken. Der Benutzer sieht so auf den ersten Blick, um welche der Lampen es sich handelt und kann sie auswählen, z.B. um sie seinem Netzwerk hinzuzufügen.

Allerdings kann diese Funktionalität - und ebenso generell alle Funktionalitäten der Inter-PAN-Befehle - auch durch Angreifer missbraucht werden. Im Identify Request wird unter anderem auch eine Dauer angegeben, für die die Lampe blinken soll. Die maximale Dauer ist nach der darin verwendeten Variable auf 18 Stunden, 12 Minuten und 14 Sekunden beschränkt. Es kann außerdem eine vom Gerät festgelegte Standarddauer (in der Regel einige Sekunden) ausgewählt werden sowie das Blinken beendet werden, indem eine Dauer von *0* verwendet wird. Ein Angreifer könnte also, ohne im Netzwerk zu sein, Lampen zum Blinken bringen und zwar für eine sehr lange Dauer.

Der Angriff wurde an verschiedenen bekannten smarten Lampen durchgeführt: Philips Hue, Osram Lightify, und GE Link. Dabei war jede der Lampen anfällig für den Blink Angriff, sobald sie an eine Stromversorgung angeschlossen war (auch wenn sie ausgeschaltet war). Während die Lampen blinken, ist es dem legitimen Benutzer nicht mehr möglich die

Lampen zu steuern. Lediglich das Trennen vom Strom oder das Aussenden des Identify Request mit der Dauer *0* durch den Angreifer beendet den Angriff.

Nach dem Angriff ging eine der Lampen wieder zurück in den Status und die Farbe, den sie zuvor hatte und die beiden übrigen setzten ihren Status und ihre Farbe zurück auf Standardwerte. Bei zwei der Lampen war die maximale Dauer des Angriffs beschränkt auf circa 9 Stunden, bei einer war der theoretische Maximalwert von etwas mehr als 18 Stunden möglich.

Der Angriff verursacht zwar keine dauerhaften Schäden oder andere Probleme, allerdings ist es für den Benutzer sehr lästig und störend, wenn seine Lampen durchgehend blinken und er diese - ohne sie vom Strom zu trennen - nicht mehr unter Kontrolle bringen kann.

3.3 Replay Angriff

Eine Erweiterung des lediglichen Abhörens und Aufnehmens der Kommunikation im Discovery Angriff ist die Verwendung der aufgenommenen Nachrichten für einen sogenannten Replay Angriff. Dieser wird sowohl in [Olawumi et al. 2014], als auch in [Saxena Sodhi Singh 2017] beschrieben.

Die zuvor aufgenommenen Nachrichten werden gespeichert und zu einem beliebigen Zeitpunkt erneut eingespielt. Damit können dann beispielsweise Befehle zum Öffnen von Türen erneut abgesendet werden, sodass ein Angreifer physikalischen Zugriff bekommen könnte. Die Möglichkeit des Replay Angriffes besteht grundsätzlich auch beim Einsatz von Verschlüsselung und ohne Kenntnis des Schlüssels, sofern die Kommunikation nach wie vor mit dem selben Schlüssel stattfindet. Allerdings kann durch Zähler, einfügen von Zeitstempeln oder sogenannte Nonces (zufällige Werte, die ebenfalls in die Nachricht eingefügt werden und nur einmal gültig sind) verhindert werden, dass eine bereits empfangene Nachricht ein weiteres Mal ausgeführt wird.

Beispiel für einen solchen Replay Angriff ist der Angriff auf ein smartes Schloss, das zum Beispiel Tresore vor unberechtigtem Zugriff schützen soll [SomersetRecon 2016]. Dabei handelt es sich um das Schloss SecuRam ProLogic B01, das mit Bluetooth Low Energy arbeitet und entweder mit einem zugehörigen am Tresor angebrachten Nummernfeld oder per Smartphone App gesteuert werden kann.

Bei der Analyse des Quellcodes nach vorherigem Dekompilieren in [SomersetRecon 2016] stellte sich heraus, dass der PIN-Code, der zum Öffnen des Schlosses vom Nutzer eingegeben werden muss, als Hexadezimalwert ausgelesen wird, nachdem er von einem String-Wert (eingelesener Text) in einen Long-Integer-Wert konvertiert wurde. Danach wird die Byte-Reihenfolge invertiert. Es findet also weder eine Entschlüsselung, noch eine Verknüpfung mit XOR oder ähnliches statt. Daraus lässt sich ableiten, dass der PIN-Code also bereits in hexadezimaler Form mit umgekehrter Reihenfolge im Datenpaket empfangen wurde. Im Quellcode konnte zwar eine Funktion gefunden werden, die eine Verschlüsselung ermöglichen könnte, diese wird jedoch nicht verwendet.

Das Mitschneiden und Analysieren der Kommunikation zwischen App und Schloss bestätigt die Erkenntnisse aus der vorherigen Code-Analyse. Es wird zum Öffnen des Schlosses ein Write-Befehl an das Schloss gesendet, wodurch bestimmte Werte geschrieben werden: Ein statischer Wert, der das Öffnen des Schlosses bezweckt, die jeweils letzten vier Byte der Sender- und Empfängeradresse, die PIN (hexadezimal in umgekehrter Reihenfolge) und die Zeit, für die das Schloss geöffnet bleiben soll. Bei jedem Ausführen des Öffnen-Befehls werden die gleichen Werte geschrieben, d.h. es findet weder eine Verschlüsselung statt, noch wird ein Zähler mit jedem Befehl hochgezählt, noch werden andere Maßnahmen ergriffen, um die Kommunikation vor Replay Angriffen zu schützen. Lediglich eine CRC-Prüfsumme

existiert, die vor Bitfehlern bei der Übertragung schützt.

Die Abbildung 4 zeigt das zweimalige Absenden eines Befehls zum Öffnen des Schlosses (Unlock-Befehl) mit der App des Herstellers. Es werden im Speziellen drei Schreibbefehle an das Schloss gesendet. Dabei ist der erste Befehl ein statischer Wert, der sich auch nicht ändert, wenn sich z.B. der PIN ändert, sodass die Autoren davon ausgehen, dass es sich dabei um den Befehl an sich (in diesem Fall das Öffnen des Schlosses) handelt. Die darauf folgenden Befehle beinhalten den PIN des Benutzers sowie die Dauer, für die das Schloss geöffnet sein soll.

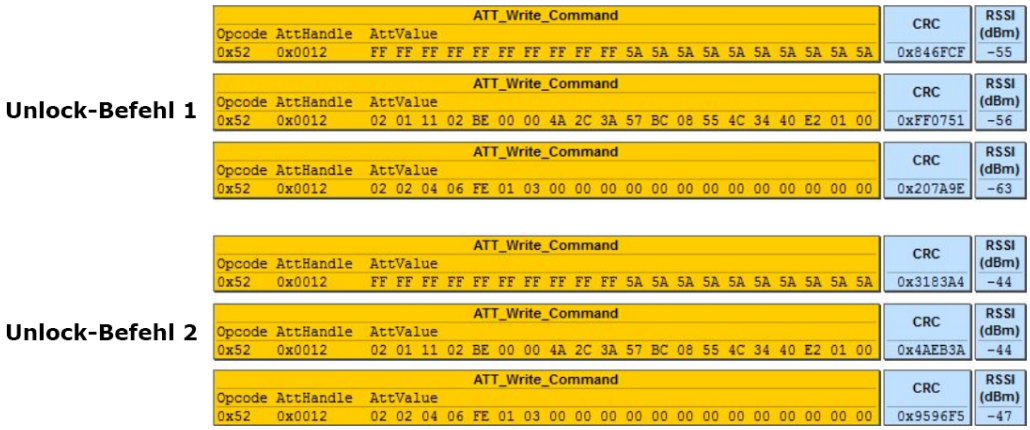

Abbildung 4: Replay Angriff bei Bluetooth Low Energy (aus [SomersetRecon 2016])

Bei erneutem Öffnen des Schlosses fällt auf, dass die gleichen Befehle erneut gesendet werden. Es sind alle Nutzdaten genau gleich, woraus offensichtlich wird, dass keine Maßnahmen gegen das erneute Einspielen von Nachrichten getroffen wurden. Ein erneutes Einspielen eines aufgezeichneten Öffnen-Befehls aus der Vergangenheit stellt also kein Problem dar und das Schloss kann ohne Weiteres geöffnet werden.

[Rose Ramsey 2016] haben ebenfalls Bluetooth Low Energy Schlösser untersucht und dabei festgestellt, dass viele smarte Schlösser anfällig für Replay Angriffe sind. Sofern keine Vorkehrungen wie die oben genannten Maßnahmen getroffen wurden, sind sowohl unverschlüsselte, als auch verschlüsselte Nachrichten vermehrbar und können für Replay-Angriffe verwendet werden.

3.4 Brute Force und Fuzzing Angriff

Bestimmte Geräte im Smart Home Umfeld wie Türschlösser erfordern die Eingabe eines Passwortes oder einer Zahlenkombination, um einen Befehl ausführen zu können. Der Benutzer gibt dieses in der App ein und kann dann ein Schloss öffnen oder schließen. Wie der Vortrag von [Rose Ramsey 2016] beschreibt, ist bei einigen Schlössern auf dem Markt jedoch das Passwort oder die Zahlenkombination auf eine bestimmte Anzahl von Zeichen begrenzt, meistens im Bereich zwischen sechs und acht Zeichen.

Manche Benutzer verwenden die Standardeinstellungen und/oder ändern Passwörter nicht oder nur sehr selten. Passwörter können per Brute Force Angriff erhalten werden. Dabei werden alle Zifferkombinationen ausprobiert, bis die richtige gefunden ist. Es kann auch sinnvoll sein, zunächst übliche einfache Ziffernfolgen wie *111111* oder *123456* auszuprobieren sowie Telefonnummern, Adressen oder ähnliches - falls bekannt - mit einzubeziehen.

Brute Force könnte nach [Celebucki Lin Graham 2018] bei BLE auch dazu verhelfen, den Short Term Key im Legacy Pairing Modus zu berechnen, bei dem der Benutzer einen PIN-

Code in beide Geräte eingibt. Auf Grund der geringen Länge (6 Zeichen) des PIN-Codes, der als temporärer Schlüssel verwendet wird, kann dieser per Brute Force erlangt werden, wenn die Kommunikation beim Schlüsselaustausch aufgenommen wurde und somit ebenfalls der Short Term Key berechnet werden.

Des Weiteren können aufgenommene Befehle - auch verschlüsselte - zufällig oder strukturiert verändert oder neue Befehle bestehend aus Zufallswerten generiert und eingespielt werden (sogenannter Fuzzing Angriff). In einem Versuch von [Rose Ramsey 2016] wurde der aufgenommene verschlüsselte Befehl zum Öffnen eines Bluetooth Low Energy Schlosses, das eine eigene (angeblich zu AES-256 ähnliche) Verschlüsselung verwendet, an bestimmten Stellen verändert und erneut abgesendet. Dies hatte zur Folge, dass das Schloss in einen Fehlerstatus überging, in dem es für den legitimen Benutzer nicht mehr kontrollierbar war. Der Benutzer bekam die Nachricht, dass die verwendeten Schlüssel veraltet sind. Im Fehlerstatus öffnet sich außerdem das Schloss, sodass ein Angreifer physikalischen Zugang zu dem vom Schloss geschützten Bereich bekommen kann.

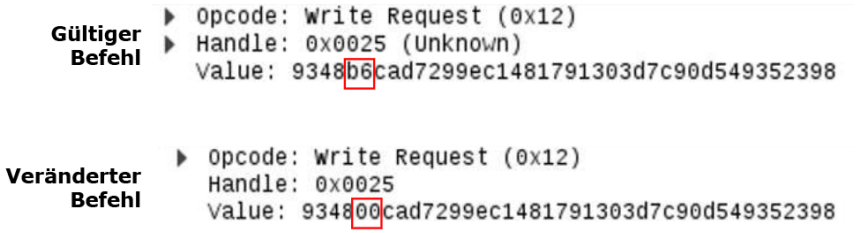

Abbildung 5: Fuzzing Angriff bei Bluetooth Low Energy (aus [Rose Ramsey 2016])

Der obere Befehl in Abbildung 5 zeigt den originalen Befehl zum Öffnen des Schlosses wie er aufgenommen wurde. Der untere Befehl ist der veränderte Befehl, der eingespielt wurde. Dabei wurde lediglich das dritte Byte zu *00* geändert, was jedoch beim Schloss für das Auslösen eines Fehlerstatus sorgte.

3.5 Energy Depletion Angriff

Geräte in der Heimautomatisierung - insbesondere kabellos kommunizierende - sind häufig mit Batterien ausgestattet, die für die Energieversorgung des Geräts sorgen. Diese sind in der Regel dafür ausgelegt das Gerät für einige Jahre zu betreiben, was auch durch energiesparende Funkprotokolle unterstützt wird. In sogenannten „Energy Depletion Attacks“ - also Angriffen auf die verfügbare Energie eines Geräts - wird versucht möglichst schnell die Batterien von außen zu leeren, sodass das Gerät ausfällt und für den Benutzer nicht mehr erreichbar ist.

Einen dieser Angriffe beschreibt die Arbeit von [Cao et al. 2016]. Es handelt sich dabei um einen Angriff auf das Funkprotokoll ZigBee und wird dort auch als „Geist in ZigBee“ (engl. „Ghost in ZigBee“) bezeichnet. Das Ziel ist es die Batterien eines bestimmten ZigBee-Geräts in einem Netzwerk möglichst schnell zu leeren, um danach weitere Angriffe durchzuführen.

Der Angriff besteht demnach aus drei Phasen: Einer Vorbereitungsphase, einer Angriffsphase und einer Phase nach dem Angriff. In der ersten Phase werden zunächst möglichst viele Informationen über die Infrastruktur gesammelt, um das Netzwerk kennen zu lernen. Dabei sollen Kenntnisse wie die Adressen von Sendern und Empfängern sowie aktive Phasen und der aktuelle Frame-Counter erlangt werden, die durch Abhören der Kommunikation und Analyse der Datenpakete gewonnen werden können (wie im zuvor beschriebenen Discovery Angriff). Der Frame-Counter wird verwendet, um das Einspielen älterer Nachrichten

zu unterbinden. Alle Nachrichten mit einem niedrigeren Wert als der aktuell auf dem Gerät gespeicherte Frame-Counter werden verworfen.

Die zuvor gewonnenen Informationen werden dann für den eigentlichen Angriff verwendet. Die Idee des Angriffs besteht darin, eine möglichst große Anzahl von selbst erstellten Nachrichten an ein Gerät zu versenden. Durch die Sicherheitsfunktionen im ZigBee-Protokoll können diese Nachrichten jedoch das betroffene Gerät nicht steuern oder Befehle ausführen lassen, denn die Nachrichten sind in der Regel verschlüsselt und der Netzwerkschlüssel ist nicht bekannt. Für den Angriff ist jedoch auch keine Kenntnis von Schlüsseln erforderlich. Des Weiteren gibt es im Optimalfall Vorkehrungen um das Verändern von Nachrichten zu erkennen (Integritätsprüfung) sowie einen Schutz vor Wiedereinspielen von bereits gesendeten Nachrichten (Schutz vor Replay Angriffen) durch einen Frame-Counter. Die erstellten Nachrichten werden allerdings alle vom Gerät empfangen und dieses muss sie zunächst verarbeiten. Dabei werden Sicherheitsberechnungen wie die Integritätsprüfung oder die Entschlüsselung durchgeführt, die durchaus Energie kosten. Da die erstellten Nachrichten die Integritätsprüfung nicht bestehen, werden sie danach verworfen. Dennoch ist bis zu dieser Stelle bereits ein Teil der Energie verbraucht worden. Mit einer großen Anzahl von falschen Nachrichten wird somit immer mehr der Energie des Geräts verbraucht.

Nachdem die Batterie des Geräts erschöpft ist, kann das Ziel des Angreifers bereits erreicht sein oder die letzte Phase des Angriffes beginnen. Das Ziel könnte erreicht sein, wenn einzelne Geräte ausgeschaltet werden sollten, z.B. sicherheitskritische Sensoren, um die Verfügbarkeit des Systems zu schwächen. Durch das permanente Senden von Nachrichten wird zudem der Funkkanal blockiert, sodass andere Geräte mit ihren Sendevorgängen warten müssen. Dadurch wird zum einen das gesamte System langsamer und zum anderen auch die Batterien der übrigen Geräte länger beansprucht.

In der letzten Phase des Angriffs können weitere Angriffe durchgeführt werden, die durch den Angriff zuvor ermöglicht werden. Nachdem die Batterie eines Geräts wieder geladen oder ausgetauscht wurde, startet das Gerät neu. Flüchtige Variablen wie der Frame-Counter sind nach dem Neustart jedoch meist zurückgesetzt auf den Wert *0*. Das hat zur Folge, dass beispielsweise in der ersten Phase aufgenommene Nachrichten wieder eingespielt werden könnten (Replay Angriff), denn die in diesen Nachrichten verwendeten Frame-Counter waren sehr wahrscheinlich höher als der nun aktuelle. Ein Angreifer könnte also Nachrichten ausführen lassen, ohne überhaupt einen verwendeten Schlüssel zu kennen. Die Abbildung 6 stellt diesen Vorgang grafisch dar.

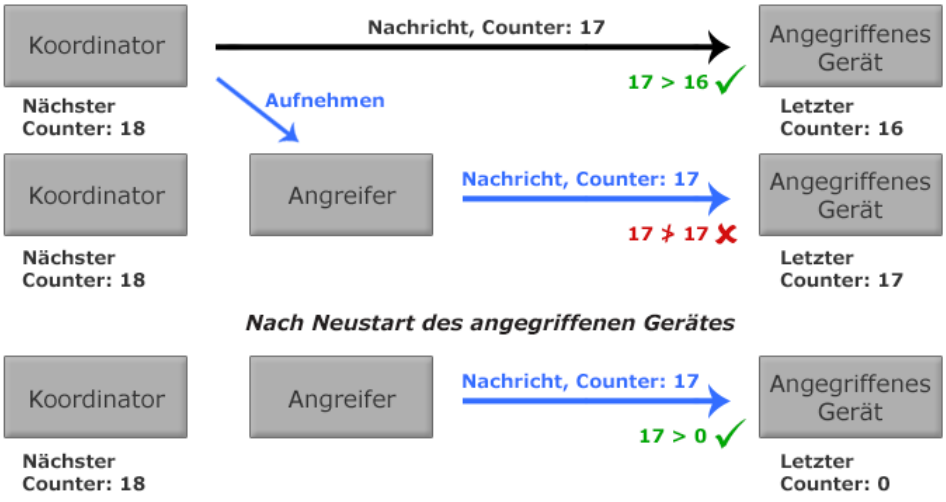

Abbildung 6: Möglichkeit eines Replay Angriffs nach Neustart eines Geräts

Durch das Wiedereinspielen der Nachrichten können zum einen unvorhersehbare Operationen ausgeführt werden, da es sich um veraltete Nachrichten handelt und zum anderen könnte es passieren, dass legitime Nachrichten (z.B. eines Koordinators) vom Gerät verworfen werden, da der in den eingespielten Nachrichten verwendete Counter als letzter Counter-Wert gespeichert wird und dieser dann höher ist als der in den legitimen Nachrichten verwendete.

Eine Gegenmaßnahme, um diesen Angriff zu verhindern oder einzuschränken, wäre die Einführung einer Liste mit Adressen, die viele fehlerhafte Nachrichten versenden (Blacklist). Nachrichten dieser Adresse könnten ab einer bestimmten Anzahl fehlerhafter Nachrichten verworfen werden und der Koordinator darüber informiert werden. Alle weiteren Nachrichten würden dann nicht mehr verarbeitet und somit käme kein hoher Energieverbrauch mehr zu Stande. Allerdings entstünden durch die Blacklist auch neue Probleme und Angriffsmöglichkeiten. Beispielsweise könnten für die fehlerhaften Nachrichten verschiedene Sendeadressen verwendet werden, sodass das betroffene Gerät früher oder später alle anderen Geräte im Netzwerk blockiert.

Eine andere Maßnahme wäre das Speichern des Frame-Counters nicht nur im flüchtigen, sondern auch in einem nicht-flüchtigen Speicher, sodass dieser beim Neustart nicht zurückgesetzt wird. Ebenfalls könnte nach dem Neustart der Netzwerkschlüssel gewechselt werden, sodass die zuvor aufgenommenen Nachrichten nicht wieder eingespielt werden können, auch wenn der Frame-Counter höher ist als der aktuelle. Diese Maßnahmen würden allerdings nicht den Energy Depletion Angriff an sich verhindern, sondern nur die Folgeangriffe nach einem Neustart.

Vergleichbare Angriffe existieren auch für andere Funkprotokolle wie Bluetooth Low Energy. [Uher Mennecke Farroha 2016] haben mit ihrem Denial of Sleep Angriff (Unterbinden des Sleep-Modus) ebenfalls das Ziel die Batterien von BLE-Geräten so stark zu beanspruchen, dass die Geräte ausgehen und nicht mehr erreichbar sind. Dabei wird ein sogenannter Sleep-Modus - ein Modus, in dem das Gerät viel Energie spart - versucht zu unterbinden, sodass das Gerät dauerhaft arbeiten muss und Energie verbraucht. Im Sleep-Modus sind nur wenige Befehle möglich und das Gerät kehrt in festgelegten Intervallen in den „normalen“ Modus zurück.

In BLE gibt es demnach verschiedene Möglichkeiten um die Batterien der Geräte möglichst schnell zu leeren. Eine der Möglichkeiten ist das schnelle und häufige Verbinden mit dem Gerät. Eine andere Möglichkeit ist es, eine große Anzahl von Schreibbefehlen auszuführen.

Die Autoren haben einen solchen Angriff auf ein Fitnessarmband (Fitbit Charge HR) ausgeführt und konnten die Batterie, die dafür ausgelegt war das Gerät für 120 Stunden mit Strom zu versorgen, mit Schreibzugriffen auf einen Sensorwert auf eine maximale Lebenszeit von 6 Stunden reduzieren.

Insgesamt kann mit Energy Depletion Angriffen die Lebenszeit von Geräten in der Heimautomatisierung deutlich verringert werden und zum anderen können durch das Ausschalten oder Neustarten der Geräte weitere Angriffe ermöglicht werden.

3.6 Key Distribution Angriff

Um das Schutzziel der Vertraulichkeit zu erreichen, kann in Netzwerken der Heimautomatisierung Verschlüsselung eingesetzt werden. In ZigBee und Bluetooth Low Energy wird dies beispielsweise mit AES-128 realisiert. Ein Empfänger ohne den richtigen Schlüssel ist zwar in der Lage verschlüsselte Nachrichten zu empfangen, kann diese jedoch nicht „lesen“, da sie für ihn lediglich eine Ansammlung von Zeichen darstellen. Mit dem richtigen Schlüssel können diese jedoch entschlüsselt werden, sodass die enthaltenen Informationen erhalten werden können.

Sogenannte Key Distribution Angriffe haben das Ziel den Schlüssel (oder ggf. mehrere Schlüssel) zu erhalten, sodass eine verschlüsselte Kommunikation entschlüsselt werden kann. Dadurch können zum einen weitere Informationen erlangt werden und zum anderen weitere Angriffe ermöglicht werden.

3.6.1 Einprogrammierte Schlüssel

Ein Beispiel für einen solchen Angriff, mit der Absicht die verschlüsselte Kommunikation in einem ZigBee-Netzwerk zu entschlüsseln, beschreibt [Yang Huang 2018]. Demnach ist der Schlüsselaustausch zwischen den beteiligten Geräten kein triviales Problem. Die Kommunikationspartner können Daten erst verschlüsselt miteinander austauschen, wenn beide einen entsprechenden Schlüssel vorliegen haben. In den gängigen Funkprotokollen der Heimautomatisierung wie ZigBee und Bluetooth Low Energy handelt es sich dabei um symmetrische Schlüssel, d.h. Sender und Empfänger verwenden die gleichen Schlüssel. Die Schwachstelle dabei ist häufig nicht das Verschlüsselungsverfahren an sich, sondern vor allem die Verteilung der Schlüssel. Diese werden beispielsweise bei ZigBee beim Eintritt eines neuen Geräts in das Netzwerk vom Koordinator an das neue Gerät übermittelt.

Die Vorgehensweise, um an den Schlüssel - im Fall von ZigBee den Netzwerkschlüssel - zu gelangen, wäre also die Netzwerkkommunikation abzuhören und mitzuschneiden. Sobald ein neues Gerät dem Netzwerk beitreten möchte, bekommt dieses den Netzwerkschlüssel mitgeteilt und kann diesen im Folgenden dann verwenden, um die verschlüsselte Kommunikation zu entschlüsseln sowie die eigene Kommunikation zu verschlüsseln.

In ZigBee gibt es zwei verschiedene Sicherheitsmodi („Standard“ oder „High-Security“), die die Art der Schlüsselverteilung spezifizieren. Im Standard-Sicherheitsmodus verschlüsselt ein Gerät im Netzwerk die Kommunikation mit dem aktuellen Netzwerkschlüssel, der entweder vorprogrammiert ist oder ungesichert vom Trust Center übertragen werden kann. Im Modus High-Security wird der übertragene Netzwerkschlüssel mit einem Masterschlüssel verschlüsselt, der auf dem Gerät bereits einprogrammiert sein kann. Ist dies nicht der Fall, wird dieser ungesichert übertragen [ZigBee 2012, S.509f.].

Vor einiger Zeit wurde jedoch bereits der fest einprogrammierte Masterschlüssel von ZigBee Light Link veröffentlicht, sodass die eigentlich verschlüsselte Übertragung des Netzwerkschlüssels, die auf der Geheimhaltung dieses Masterschlüssels beruhte, zumindest bei ZigBee Light Link nicht mehr sicher ist [Sokolov 2015].

Nach [Zillner 2015] wird beim ZigBee Profil für die Heimautomatisierung (Home Automation Public Application Profile, HAPAP) für die Kommunikation mit dem Trust Center im Netzwerk, das beim Beitritt eines neuen Geräts den Netzwerkschlüssel übermittelt, ein Standardschlüssel (Default Trust Center Link Key) zur Verschlüsselung des Netzwerkschlüssels verwendet. Dieser ist im Standard festgelegt als Hexadezimalwert von „ZigBeeAlliance09“ im Little-Endian Format.

Bei Bluetooth Low Energy ist insbesondere bei der Pairing-Methode „Just Works“, bei der der temporäre Schlüssel mit Nullen verschlüsselt übertragen wird, ein Mitlesen möglich. Mit diesem Schlüssel kann dann der Short Term Key berechnet werden.

Nachdem die Kommunikation mit den erhaltenen Schlüsseln entschlüsselt werden kann, sind weitere Tätigkeiten und auch weitere Angriffe möglich. So können beispielsweise Daten wie Befehle an bestimmte Geräte analysiert werden oder eigene Befehle erstellt werden, mit denen Geräte gesteuert und z.B. auch aus dem Netzwerk entfernt werden können. Außerdem könnten Daten abgefangen und manipuliert werden [Yang Huang 2018].

3.6.2 Jamming Angriff

Die Arbeit von [Zillner 2015] beschreibt Schwachstellen beim Protokoll ZigBee, insbesondere beim Austausch bzw. bei der Verteilung des Netzwerkschlüssels, mit dem sämtliche Kommunikation im ZigBee Netzwerk verschlüsselt ist. Demnach werden bei ZigBee - wie in Punkt 3.6.1 bereits beschrieben - fest einprogrammierte Schlüssel verwendet, um den Netzwerkschlüssel an neu beitretende Geräte zu verteilen. Falls das Gerät jedoch keine einprogrammierten Schlüssel besitzt, kann der Schlüssel ersatzweise (als „Fallback“) unverschlüsselt übertragen werden. Somit bleibt für den Angreifer ein kleiner Zeitraum, um den Schlüssel beim Eintritt des Geräts ins Netzwerk mitzuschneiden - entweder unverschlüsselt oder mit einem festgelegten Schlüssel verschlüsselt. Im Fall von ZigBee können diese Schlüssel bereits veröffentlicht sein, sodass der Angreifer von ihnen bereits Kenntnis haben kann.

Ein Angreifer kann jedoch auch aktiv tätig werden um an den Schlüssel zu gelangen, indem er Störgeräusche einspielt (sogenanntes „Jamming“). Ein typischer Besitzer der Geräte könnte durch die fehlerhafte oder nicht mehr mögliche Kommunikation seines Geräts dazu gebracht werden, dass dieser sein Gerät auf den Werkszustand zurücksetzt (Factory Reset) oder das Gerät neu mit dem Netzwerk versucht zu verbinden. Denn durch das Aussenden von Störgeräuschen durch den Angreifer wirkt das Szenario für einen „normalen“ Benutzer wie ein gewöhnlicher Verbindungsverlust zum Netzwerk beispielsweise auf Grund eines zu großen Abstandes zwischen Gerät und Koordinator. Bei den Störgeräuschen muss es sich nicht um gültige ZigBee Nachrichten handeln, die den Kanal für alle weitere legitime Kommunikation blockieren, sondern es kann sich um beliebige Daten (Rauschen) handeln, sodass die Kommunikation gestört wird.

Der Benutzer initiiert also erneut das Übertragen des Netzwerkschlüssels vom TrustCenter bzw. von der Zentrale im Allgemeinen an das neue Gerät und der Angreifer bekommt erneut die Chance den Schlüssel mitzuschneiden. Durch die ungesicherte Übertragung oder durch die Verwendung von bekannten Schlüsseln können Angreifer den Netzwerkschlüssel mitlesen und die Kommunikation im Folgenden entschlüsseln.

3.6.3 Man-in-the-Middle Angriff

Um an sensible Daten wie den Netzwerkschlüssel zu gelangen oder die Kommunikation mitzuhören, kann auch ein sogenannter Man-in-the-Middle Angriff (kurz MITM) erfolgreich sein, der für Bluetooth in [Albahar 2017] beschrieben wird. Bei dieser Art von Angriff schaltet sich der Angreifer zwischen die Kommunikation zweier Geräte und kommuniziert mit diesen. Die Geräte haben den Eindruck sie kommunizieren mit dem jeweils anderen Gerät, in Wirklichkeit kommunizieren sie jedoch beide mit dem Gerät des Angreifers. Somit kann dieser Pakete entweder unverändert weiterleiten, sie löschen oder aber auch verändern. Der MITM Angriff ist grafisch in Abbildung 7 dargestellt.

Wie in der Abbildung dargestellt ist, versuchen zunächst beide Teilnehmer sich mit dem jeweils anderen zu verbinden. Beispielsweise durch Senden von Störsignalen kann es aber einem Angreifer gelingen, dass sein Gerät, das sich aus Sicht der beiden Teilnehmer als der jeweils andere ausgibt, als Kommunikationspartner fungieren kann. Somit verbindet sich Gerät A mit dem Gerät des Angreifers und Gerät B ebenfalls. Da alle Nachrichten nun über das Gerät des Angreifers abgewickelt werden, kann dieser beispielsweise statt der Weiterleitung der Information, dass das Gerät über eine Ein- und Ausgabe-Funktionalität verfügt, übermitteln, dass weder Ein- noch Ausgabe möglich ist. In diesem Fall verbinden sich also die Geräte per Pairing-Methode „Just Works“, die anfällig für MITM Angriffe ist. Somit wird der Schlüssel, der lediglich mit Nullen verschlüsselt ist, an das Gerät des Angreifers übertragen. Dieser schließt mit beiden angegriffenen Geräten den Pairing-Vorgang ab und

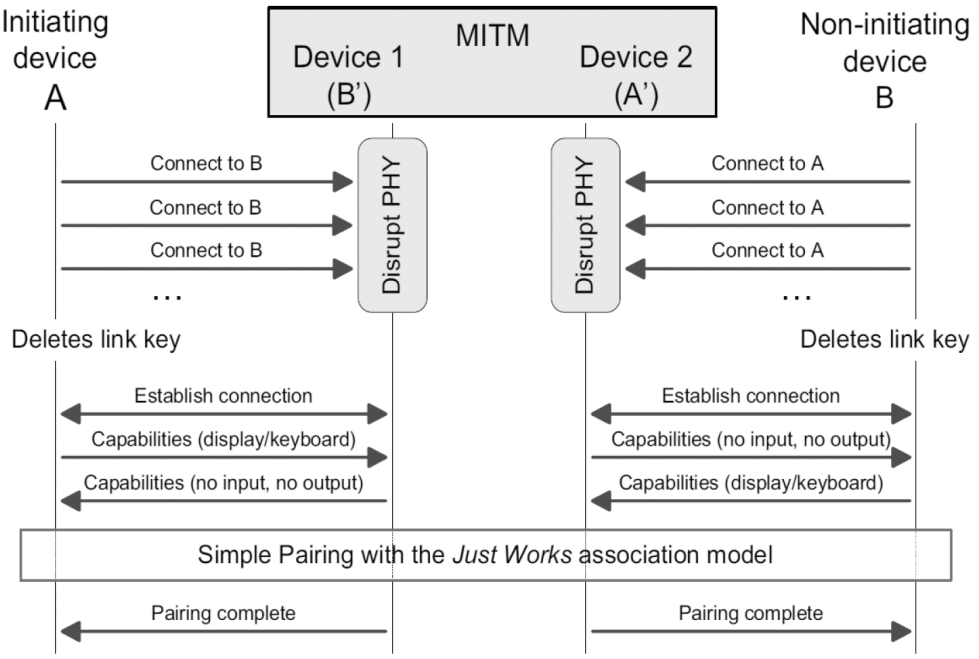

Abbildung 7: Man-in-the-Middle Angriff auf Bluetooth Low Energy (aus [Albahar 2017])

kann somit die gesamte Kommunikation mitlesen.

Die Autoren von [Ray et al. 2018] haben einen Man-in-the-Middle Angriff an drei verschiedenen Sensoren, die BLE verwenden, erfolgreich ausführen können.

3.6.4 Security Downgrade Angriff

Ein ähnlicher Angriff wie der im ersten Unterpunkt beschriebene Key Distribution Angriff mit fest einprogrammierten Schlüsseln ist der Security Downgrade Angriff in Z-Wave. In dem in [Tierney 2018b] vorgestellten Angriff geht es um das Ausnutzen der Abwärtskompatibilität beim Verbinden zweier Geräte. Beim Pairing-Prozess zwischen einem Z-Wave Controller und einem neuen Gerät wird der Netzwerkschlüssel ausgetauscht.

In einem veralteten Pairing-Prozess „S0“ wird bei älteren Geräten der Netzwerkschlüssel mit einem Schlüssel bestehend aus Nullen verschlüsselt übertragen. Da diese Übertragung unsicher ist, unterstützen neuere Geräte den neuen Prozess „S2“. Bei diesem wird ein Schlüsselaustausch mit dem Diffie-Hellman Algorithmus durchgeführt und zusätzlich durch die Eingabe eines 5-Zeichen langen Codes durch den Benutzer authentifiziert.

Das Diffie-Hellman-Verfahren schützt vor dem Mitschneiden der Kommunikation beim Schlüsselaustausch, indem beide Kommunikationspartner Geheimnisse besitzen, die sie nicht übertragen. Mit diesen und mit Hilfe des diskreten Logarithmus, wird ein Wert berechnet, den beide Partner austauschen. Dieser ist nur mit viel Rechenaufwand umkehrbar. Daraus können beide dann einen gemeinsamen symmetrischen Schlüssel berechnen [Heise 2018].

Allerdings müssen alle Z-Wave Geräte abwärtskompatibel sein und weiterhin den S0-Prozess beherrschen. Welcher der beiden Prozesse verwendet wird, bestimmt eine vom neuen Gerät ausgesendete Node-Info, die Informationen enthält, welche Funktionalitäten das Gerät mitbringt. Ist es in der Lage den S2-Prozess auszuführen, teilt es dies mit, ansonsten wird der unsichere S0-Prozess verwendet. Die Node-Info ist eine weder verschlüsselte, noch authentifizierte Nachricht und ist somit relativ einfach unbemerkt manipulierbar. Sie wird lediglich durch eine Prüfsumme vor Übertragungsfehlern geschützt.

Die Idee dieses Angriffs in [Tierney 2018b] ist es den Pairing-Prozess so zu beeinflussen, dass die unsichere S0-Variante verwendet wird und der Schlüssel somit relativ einfach mitgelesen und später verwendet werden kann.

Der Pairing-Prozess kann eingeleitet werden, indem der Benutzer einen Knopf sowohl auf dem Controller als auch auf dem neuen Gerät betätigt. Das Drücken des Knopfes auf dem Gerät bewirkt das Aussenden der Node-Info. Der Knopf des Controllers bewirkt das Erwarten dieser und Einleiten des Verbindens. Ein Angreifer könnte die Node-Info abfangen, dahingehend verändern, dass die Information über die Möglichkeit von S2 entfernt wird und an den Controller weiterleiten, sodass dieser gezwungen ist den S0-Prozess zu verwenden. Bei dieser Weiterleitung muss das Gerät des Angreifers nur die gleiche Home-ID wie die des neuen Geräts mitsenden, die jedoch abgehört werden kann.

In [Tierney 2018b] wurde ein smartes Türschloss verwendet, um den Angriff durchzuführen. Bei erfolgreichem Erhalt des Schlüssels könnte dieses vom Angreifer dann gesteuert werden, sodass physikalischer Zugang beispielsweise zu einer Wohnung möglich wird.

Die Manipulation der Node-Info kann nach drei verschiedenen Varianten erfolgen: In der ersten Variante drückt der Benutzer als erstes die Taste auf dem neuen Gerät und dann erst die Taste auf dem Controller. Somit bleibt ein sehr kurzer Zeitraum für den Angreifer die Node-Info aufzunehmen, den S2-Teil daraus zu entfernen und diese nach Drücken der Taste am Controller einzuspielen. Bei der zweiten Variante senden manche Geräte bereits Node-Infos aus, sobald sie mit dem Strom verbunden werden. Somit könnte ein Angreifer diese bereits speichern und verändern. Sobald die Taste am Controller gedrückt wird, kann er diese veränderte Node-Info senden, sodass hier noch nicht einmal eine Interaktion des Benutzers mit dem neuen Gerät notwendig ist. Die dritte Variante setzt das Drücken der Tasten in der richtigen Reihenfolge (erst Controller, dann neues Gerät) voraus. Der Angreifer hört das Aussenden der Node-Info mit und sobald er die Home-ID des Geräts empfangen hat, beginnt er ein Störsignal auszusenden, das es verhindert, dass die Node-Info beim Controller ankommt. Dann sendet er selbst eine veränderte Node-Info ohne den S2-Teil ab.

Alle drei Varianten führen dazu, dass der Controller auf Grund der fehlenden Möglichkeit für den S2-Prozess beim neuen Gerät den unsicheren S0-Prozess wählt und beide Geräte sich mit diesem Verbinden. Der Angreifer kann dann den mit Nullen verschlüsselten Netzwerkschlüssel mithören und für weitere Angriffe verwenden.

3.7 Reset Angriff

Ein weiterer Angriff, der in [Ronen et al. 2017] vorgestellt wird, nutzt Implementierungsfehler in den bekannten Philips Hue (smarte Lampen) aus, die das ZigBee Protokoll - insbesondere das für Beleuchtungssteuerung vorgesehene ZigBee Light Link - verwenden, um eine Kettenreaktion auszulösen. Der Angriff hat zum Ziel die angegriffenen Geräte - in diesem Fall Lampen - auf ihren Werkszustand zurückzusetzen. Dieses Zurücksetzen hat dann zur Folge, dass die Geräte darauf warten wieder einem ZigBee-Netzwerk beizutreten, das auch das Netzwerk eines Angreifers sein kann. Dadurch kann dieser die volle Kontrolle über das Gerät bekommen und der ursprüngliche Besitzer ist nicht mehr in der Lage dieses zu steuern.

Der beschriebene Angriff besteht nicht nur aus der Übernahme der Lampen in ein Netzwerk des Angreifers, sondern geht noch einen Schritt weiter und übermittelt bösartige Firmware-Updates auf die Geräte, die dafür sorgen, dass diese sich gegenseitig mit diesen Updates infizieren. Zunächst wird jedoch der erste Teil des Angriffs, nämlich der Reset Angriff, beschrieben.

Bei ZigBee Light Link (ZLL) können Geräte neu in das Netzwerk aufgenommen werden, indem das sogenannte „Touchlink Commissioning“ verwendet wird. Dabei handelt es sich

um das Verfahren, das bereits im Punkt 3.6 beschrieben wurde. Einem neu ins Netzwerk beitretenden Gerät wird der Netzwerkschlüssel verschlüsselt mit einem Masterschlüssel mitgeteilt. Durch das Touchlink Commissioning soll die Benutzbarkeit für die Benutzer stark erhöht werden, indem der Verbindungsprozess zwischen einem neuen Gerät und dem Netzwerk vereinfacht wird. Diese Vereinfachung findet so statt, dass ein Gerät sich lediglich in einem sehr geringen Radius (weniger als ein Meter) um ein Gerät, mit dem es sich verbinden soll, befinden muss, um Befehle auszutauschen. Dieser geringe Abstand soll dazu dienen, dass nicht jedes beliebige Gerät aus der Ferne gesteuert werden kann, denn jede Nachricht eines Geräts, das weiter entfernt ist, soll laut ZigBee Standard verworfen werden. Wie weit ein Gerät entfernt ist, wird mittels der Signalstärke einer Nachricht beim Empfänger gemessen. Nur wenn diese einen bestimmten Wert überschreitet, wird die Nachricht angenommen und verarbeitet. Da die Empfangsleistung aus physikalischen Gründen quadratisch mit der Entfernung abnimmt, wird so auch mit der Verwendung von Geräten mit stärkerer Sendeleistung die Befehlsabgabe aus größeren Entfernungen deutlich erschwert.

Der erste Schritt des Touchlink Commissioning ist das Senden eines Befehls „Scan Request“ an das neue Gerät, das dem Netzwerk beitreten soll. Befindet sich der Sender in näherer Umgebung (Empfangssignal überschreitet festgelegten Schwellwert), dann wird entsprechend ein Befehl „Scan Response“ vom Empfänger zurückgesendet. In diesen beiden Befehlen werden sowohl eine 32-Bit große Transaktions-ID (ungleich *0*), als auch eine 32-Bit große Response-ID ausgetauscht, die beide Geräte dann im Folgenden verwenden können, um die Transaktionen eindeutig zu identifizieren.

Mit einer gültigen Transaktions-ID sind zwei Befehle möglich, die für einen Angreifer interessant sein können: Ein „Reset To Factory New Request“ lässt das Gerät wieder in den Werkszustand zurückkehren, d.h. es werden sowohl alle Schlüssel als auch alle Informationen über das aktuelle Netzwerk gelöscht. Außerdem kann mit dem Befehl „Join Network Request“, dem auch Informationen über die PAN-ID eines neuen Netzwerkes und der Netzwerkschlüssel verschlüsselt mit dem veröffentlichten ZLL-Masterschlüssel übergeben werden, das Gerät dazu gebracht werden einem neuen Netzwerk beizutreten.

Damit nicht alle Geräte, die den ZLL-Masterschlüssel besitzen - und das sind alle ZigBee-zertifizierte Geräte - jedes beliebige Gerät zurücksetzen oder in ein anderes Netzwerk hinzufügen kann, wird die oben angesprochene Abstandsprüfung (engl. Proximity Check) durchgeführt. Bei Philips Hue liegt der erlaubte Abstand um diese Befehle auszuführen unterhalb von einem Meter, d.h. das Gerät, das zurückgesetzt oder mit einem neuen Netzwerk verbunden wird, darf sich maximal einen Meter im Umkreis des Senders der Befehle aufhalten. Nur dann wird die Nachricht entgegengenommen und ausgeführt.

Allerdings gibt es in der Implementierung der Philips Hue einige sicherheitsrelevanten Probleme, wodurch ein Reset Angriff überhaupt erst möglich wird. Denn durch die ZigBee Spezifikation an sich ist ein solcher Angriff nur möglich, wenn der Angreifer sich im Umkreis von unter einem Meter um das anzugreifende Gerät befindet, einen Scan Request ausgeführt hat und den Befehl zum Zurücksetzen mit einer gültigen Transaktions-ID aus dem Scan Request sendet.

In Philips Hue wird der Proximity Check beim Empfangen eines Scan Request ausgeführt. Zudem ist die Transaktions-ID *0* hier nicht gültig und ein Scan Request mit dieser Transaktions-ID wird verworfen. Die gültigen Transaktions-IDs werden in einem Array gespeichert, das zunächst mit Nullen befüllt ist, da dies ohnehin keine gültige Transaktions-ID ist. Beim Empfangen eines Befehls zum Zurücksetzen des Geräts wird zunächst geprüft, ob die Transaktions-ID gültig ist. Denn eine gültige Transaktions-ID kann nur existieren, wenn das Gerät erfolgreich einen Scan Request gesendet hat (was wiederum bedeutet, dass sich das Gerät in der direkten Umgebung befunden haben muss). Jedoch wird beim Empfangen eines anderen Befehls als eines Scan Requests die Funktion, die die Entfernung prüft, nicht

mehr aufgerufen. Das bedeutet, dass zwar ein Scan Request - wie es in der Spezifikation vorgesehen ist - verarbeitet wird, jedoch alle weiteren Befehle auch aus weiteren Entfernungen durchgeführt werden können. Zudem wird bei allen weiteren Befehlen auch nicht mehr geprüft, ob die Transaktions-ID *0* ist. Es wird lediglich geprüft, ob die Transaktions-ID gültig ist und da das Array mit gültigen Transaktions-IDs standardmäßig mit Nullen befüllt ist, ist die Transaktions-ID *0* gültig.

Die Abbildung 8 zeigt den Prozess der Abstandsprüfung bei Philips Hue. Aus Gründen der Übersichtlichkeit sind in der Grafik die Transaktions-ID (TID) sowie die Response-ID (RID) nicht als 32-Bit Werte dargestellt.

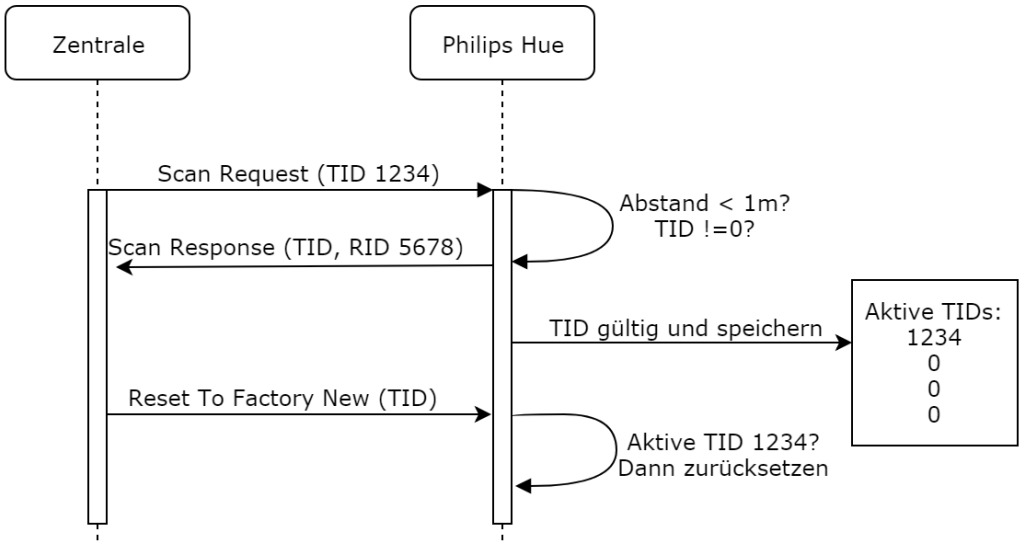

Abbildung 8: Abstandsprüfung in Philips Hue (nach [Ronen et al. 2017])

Somit wird es möglich, einen Befehl „Reset To Factory New“ mit der Transaktions-ID *0* zu senden, der aus beliebiger Entfernung akzeptiert und ausgeführt wird. Dies ist in Abbildung 9 dargestellt.

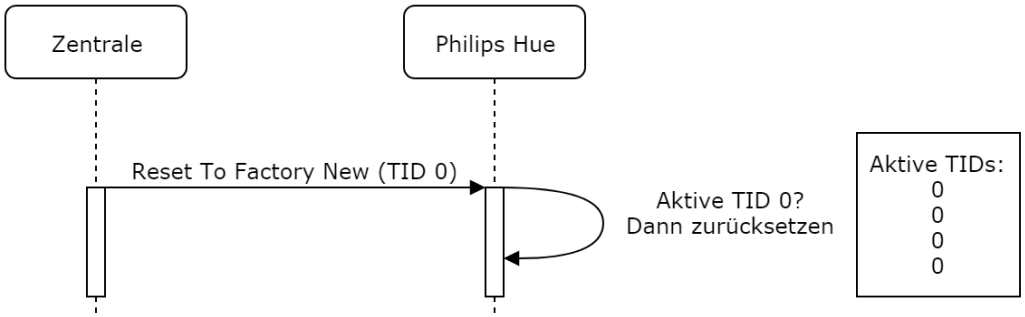

Abbildung 9: Reset Angriff bei Philips Hue (nach [Ronen et al. 2017])

Nachdem die Lampen dann zurückgesetzt wurden, suchen Sie per Aussendung des Befehls „Beacon Request“ in einem bestimmten Intervall nach Netzwerken, denen sie beitreten sollen. Ein Angreifer könnte diese Nachrichten also beantworten und die Geräte so zu seinem Netzwerk hinzufügen, sodass der ursprüngliche Besitzer keine Kontrolle mehr über die Geräte hat.

Ein weiterer Implementierungsfehler ermöglicht zusätzlich eine Verbesserung des Angriffs. Laut ZigBee Standard sollten solche Befehle zum Zurücksetzen auf den Werkszustand eines

Geräts lediglich die Adresse eines einzelnen Geräts, das angesprochen werden soll, enthalten. In den Lampen von Philips werden jedoch keine Maßnahmen ergriffen die Adressen zu prüfen, sodass auch Broadcast-Adressen als Ziel möglich sind. So können also nicht nur einzelne Lampen angesprochen werden, sondern gleich alle Lampen im Netzwerk, die auf die Broadcast-Adresse reagieren. Somit kann in kurzer Zeit eine Vielzahl von Lampen übernommen werden.

Da der eigentliche Besitzer der Lampen in der Regel physikalischen Zugang zu seinen Lampen haben sollte, wäre er jedoch in der Lage die Lampen wieder zurück in sein Netzwerk zu bringen. Der Schaden wäre also nur ein temporärer, bedeutet jedoch einen Aufwand für den legitimen Besitzer seine Lampen wieder zurückzuholen.

Zusammenfassend ist es möglich - zumindest bei Philips Hue Lampen - mittels Touchlink Commissioning von ZigBee Light Link, das eigentlich für mehr Benutzerkomfort beim Einbinden neuer Geräte ins Netzwerk sorgen soll, in Kombination mit Implementierungsfehlern Geräte aus der Ferne zurückzusetzen, sie in das eigene (Angreifer-) Netzwerk einzubinden und so die Kontrolle über sie zu bekommen.

3.8 Firmware-Update Angriff

Der Reset Angriff aus 3.7 wird von [Ronen et al. 2017] noch weitergeführt, um eine Kettenreaktion auszulösen, wodurch außerdem auch dauerhafter Schaden an den Geräten angerichtet werden kann. Die Idee dabei ist es, die Lampen, nachdem sie übernommen wurden und sich im Netzwerk des Angreifers befinden, mit einem Firmware-Update zu bespielen, das weitere Lampen im Umkreis infiziert.

Geräte der Heimautomatisierung können eine sogenannte Over-The-Air (OTA) Funktion bereitstellen, mit der der Hersteller Firmware-Updates auf die Geräte einspielen kann, um diese auf dem aktuellen Stand zu halten. Das Verfahren der Übermittlung ist ebenfalls standardisiert. Allerdings definiert der Standard lediglich ein Client-Server Protokoll für die Übermittlung und empfiehlt, dieses asymmetrisch zu verschlüsseln. Für die Hersteller ist die Verwendung einer asymmetrischen Verschlüsselung der Updates jedoch nicht verpflichtend. Die Verschlüsselung der Updates soll dafür sorgen, dass diese sowohl authentifiziert als auch unverändert und fehlerfrei sind.

Bei den Philips Hue Lampen wird ebenfalls eine OTA-Funktionalität zur Verfügung gestellt, mit dem die Geräte Updates erhalten können. Diese sind mittels der symmetrischen Verschlüsselung AES-CCM verschlüsselt. Der Schlüssel ist dabei zumindest für jedes Gerät eines spezifischen Modells gleich.

Damit ein Update also von einem Gerät angenommen werden kann, muss es mit dem gültigen OTA-Schlüssel verschlüsselt sein. Da es sich bei den Lampen um eine symmetrische Verschlüsselung handelt, wird der Schlüssel nicht nur für die Entschlüsselung nach Erhalt des Updates am Gerät verwendet, sondern auch von den Herstellern für die Verschlüsselung der Updates. Das bedeutet, dass Angreifer, die in Besitz des Schlüssels kommen, dann auch in der Lage sind eigene Updates herzustellen, die dann von den Geräten akzeptiert werden.

In [Ronen et al. 2017] wird der verwendete Schlüssel durch eine Seitenkanalanalyse extrahiert. Eine solche Analyse misst beispielsweise den Energieverbrauch eines Geräts bei der Durchführung von Rechenoperationen für die Verschlüsselung. Dieser steht in einem Zusammenhang mit den Operationen und dem verwendeten Schlüssel. Somit können Informationen über den Energieverbrauch Rückschlüsse auf den verwendeten Schlüssel geben.

Die Extraktion des Schlüssels war zwar mit einem etwas größeren Aufwand verbunden, die Autoren vermuten jedoch, dass dieser Schlüssel in Zukunft - ähnlich wie der ZLL-Masterschlüssel - öffentlich werden kann und die Problematik mit Firmware-Updates dadurch noch

größer wird.

Mit dem erhaltenen OTA-Schlüssel können dann vorhandene legitime Updates entschlüsselt, bearbeitet und verschlüsselt werden, die dann auf Geräte eingespielt werden können. Um eine Kettenreaktion auszulösen, die das Ziel des gesamten Angriffes sein soll, kann also das Update so verändert werden, dass jedes Gerät seine Nachbarn im Umfeld ebenfalls infiziert. Letztendlich ist es ebenfalls möglich, zukünftige Updates zu unterbinden, sodass der Firmware-Update Angriff irreversibel ist und die infizierten Geräte unbrauchbar werden.

Damit Updates vom Gerät erkannt werden, wird auf dem Flash-Speicher des Geräts ein Flag gesetzt, das angibt, ob ein Update verfügbar ist und das Update wird an einer bestimmten Stelle auf dem Flash-Speicher abgelegt. Beim Start des Geräts wird das Update dann eingelesen und entschlüsselt sowie verifiziert. Um die OTA-Upgrade Funktionalität zu verwenden, muss der Angreifer jedoch zunächst den Netzwerkschlüssel besitzen, da die Updates ebenfalls verschlüsselt vom Update-Server (kann beispielsweise der Koordinator sein) übertragen werden [ZigBee 2014, S.19]. Der Netzwerkschlüssel kann jedoch mit der Methode aus Abschnitt 3.6 erhalten werden oder der Netzwerkschlüssel ist bekannt, da das Gerät, nachdem es mit der Methode aus Abschnitt 3.7 zurückgesetzt wurde, dem Angreifernetzwerk beigetreten ist und den Schlüssel dieses Netzwerkes verwendet. Da die Update-Befehle ebenfalls nicht der Scan Request Befehl ist, wird auch hier bei den Philips Hue Lampen keine Abstandsprüfung durchgeführt [Ronen et al. 2017].

Zusammenfassend ist es also möglich - mit etwas höherem einmaligem Aufwand den symmetrischen Schlüssel der Updates bestimmter Gerätegruppen zu erhalten - eigene Updates zu erstellen bzw. vorhandene Updates zu verändern, die schädlichen Code beinhalten. Diese können dann per OTA-Funktionalität auf Geräte übertragen werden und werden dort ausgeführt, womit es auch möglich ist eine Kettenreaktion auszulösen, indem weitere Lampen in der Umgebung infiziert werden.

3.9 Decompile Angriff

In der Veröffentlichung von [Tierney 2018a] werden Sicherheitslücken in einem smarten Vorhängeschloss (dem „Tapplock Smart Lock“) aufgezeigt, das mit Bluetooth Low Energy gesteuert wird. Das Schloss kann entweder durch einen Fingerabdruckscanner geöffnet werden, der am Schloss selbst angebracht ist, oder durch eine App. Dabei fiel auf, dass bei jedem Verbinden mit dem Schloss der gleiche Write-Befehl an das Schloss gesendet wird. Dieser sieht aus wie eine Aneinanderkettung zufälliger Zeichen - könnte also auf eine Verschlüsselung hinweisen.

In diesem Fall kam ein sogenannter Decompile Angriff zum Einsatz. In diesem wird der Kompiliervorgang bei der Fertigstellung einer Software wieder rückgängig gemacht. Bei der Software kann es sich entweder um die Firmware der Hardware handeln - was jedoch unter Umständen problematisch sein könnte, da diese verschlüsselt sein kann. Es kann sich aber auch um die Software handeln, mit der der Benutzer das Gerät steuern können soll, also beispielsweise eine App. Aus der kompilierten Software wird mit bestimmten Tools wieder lesbarer Code erzeugt, sodass dieser analysiert werden kann, um zu verstehen welche Mechanismen genau angewandt werden.

Welche Software in [Tierney 2018a] verwendet wurde, um die App zu dekompilieren, ist nicht bekannt. Es existieren jedoch viele verschiedene Tools, u.a. auch webbasiert und kostenlos, die diese Funktionalität beherrschen und je nach Programmiersprache zum Einsatz kommen können.

Im Fall von [Tierney 2018a] wurde die App des Schlosses dekompiliert. Nach einer Analyse des Codes, insbesondere beim Verbinden der App mit dem Schloss, ergab sich die Erkenntnis,

dass die empfangenen Befehle keine verschlüsselten Befehle sind. Es handelt es sich um eine Variable „Key1“ sowie eine Variable „SerialNo“, die vom Benutzer übertragen werden müssen, um das Schloss zu steuern.

Diese beiden Variablen beinhalten jedoch keine Verschlüsselung. Vielmehr wird mit der MAC-Adresse des Schlosses ein MD5-Hashwert gebildet und die ersten acht Zeichen dieses Hashwertes bilden die Variable „Key1“, die Zeichen 16 bis 23 ergeben die Variable „SerialNo“. Um sich mit dem Schloss zu verbinden und dieses zu öffnen, wird also lediglich die MAC-Adresse des Schlosses benötigt. Hier gibt es also keinerlei Sicherheit durch Verschlüsselung oder ähnliche Mechanismen, auch wenn die gesendeten Befehle zunächst diesen Eindruck vermitteln.

Da die MAC-Adresse des Schlosses immer gleich ist, ist auch jeder Befehl immer gleich und somit ist das Schloss ebenfalls anfällig für Replay Angriffe.

Durch die Analyse der App, die dekompiliert und der Code so lesbar wurde, konnten also wichtige Informationen über die Befehle erlangt werden. Zwar wäre es auch per Replay Angriff bereits möglich gewesen das Schloss zu öffnen, ein Decompile Angriff lässt jedoch erst die Erkenntnisse zu, was genau für Befehle gesendet werden. Für einen Außenstehenden sehen diese unter Umständen zunächst wie verschlüsselte Befehle aus, bei der Analyse des Codes konnte eine Verschlüsselung jedoch nicht entdeckt werden.

3.10 Sybil Angriff

In der Arbeit von [Jasek 2016] wird außerdem beschrieben, dass es mit Fälschen bzw. Klonen von Geräten möglich sein kann, vorzutäuschen es handele sich bei Geräten eines Angreifers um legitime Geräte. Das Klonen von Geräten und Fälschen dessen Identität, um in dessen Namen im Netzwerk zu agieren, wird in [Deogirikar Vidhate 2017] auch als Sybil Attack bezeichnet.

Für Bluetooth Low Energy wird in [Jasek 2016] beschrieben, dass es bereits ausreichend sein kann Beacon-Daten (Daten, die ein BLE Gerät aussendet, damit andere sich mit ihm verbinden können) zu manipulieren. Darin wird auch die MAC-Adresse des Geräts angegeben, die ein Angreifer fälschen bzw. klonen kann. Die Anwendungen des Benutzers identifizieren ein Gerät unter Umständen nur daran und bei einem häufigeren Aussenden (höheren Intervall) der Beacon-Nachrichten wird das erste Gerät, das eine Beacon-Nachricht sendet, zum Verbinden ausgewählt. Somit wird das legitime Gerät vom Benutzer nicht mehr gesteuert, sondern das des Angreifers. Dadurch kann es zu Problemen mit der Verfügbarkeit des Geräts kommen, das der Benutzer ursprünglich steuern wollte.

Des Weiteren können auch andere Werte in den Beacon-Nachrichten verändert werden, wie z.B. der Gerätestatus (Status ON obwohl das Gerät im Status OFF ist), sodass durch die Anwendung des Benutzers Befehle abhängig von diesem Status gesendet werden und dadurch Fehler auftreten.

Ebenfalls wird beschrieben, dass auch ganze Geräte geklont werden können. Dafür wird die selbe MAC-Adresse auf dem Gerät des Angreifers vorgetäuscht, die auch das originale Gerät hat sowie die gleichen Services und Characteristics implementiert (mit den gleichen Handle-IDs). Bei häufigerem Aussenden der Beacon-Nachrichten finden die Geräte eher das Gerät des Angreifers und verbinden sich mit diesem. Dadurch kann entweder ebenfalls ein Problem mit der Verfügbarkeit auftreten oder aber der Angreifer kann so an Daten kommen, die ihm gesendet werden, aber eigentlich nicht für ihn bestimmt sind.

In der Arbeit von [Hassan et al. 2017] wird der Sybil Angriff auch als „MAC Address Spoofing Attack“ für Bluetooth beschrieben. Die Abbildung 10 stellt den Sybil Angriff grafisch dar.

Abbildung 10: Sybil Angriff bei Bluetooth (aus [Hassan et al. 2017])

Demnach sind zwei Bluetooth-Geräte in der Vergangenheit miteinander verbunden gewesen. Sie haben also bereits einen gemeinsamen Schlüssel ausgehandelt und miteinander kommuniziert (Punkt 1 in der Grafik). Ein Angreifer hat Kenntnis über die MAC-Adresse eines der Geräte (z.B. durch Mitschneiden der Kommunikation) und ist in der Lage das Gerät zu klonen (Punkt 2 und 3 in der Grafik). Das Gerät des Angreifers täuscht nun vor, das legitime Gerät zu sein, mit dem sich das Gerät zuvor bereits verbunden hat, indem es mit der MAC-Adresse des legitimen Geräts agiert.

Wenn die Kommunikation zuvor verschlüsselt stattgefunden hat, haben beide legitimen Geräte die Schlüssel, die ausgetauscht wurden, noch gespeichert und können diese direkt verwenden, ohne einen neuen Schlüssel aushandeln zu müssen. Wenn sich das angegriffene Gerät jedoch mit dem Gerät des Angreifers verbindet, ist dieser Schlüssel beim Angreifer nicht vorhanden, sodass das Gerät dazu gebracht wird, sich neu zu verbinden, d.h. mit dem Angreifer einen neuen Schlüssel auszuhandeln (sogenannter „Forced Re-Pairing“ Angriff). Dadurch ist der Angreifer nun mit einem legitimen Gerät verbunden, das den Angreifer für ein anderes legitimes Gerät hält und somit Daten übermitteln könnte, die nicht für den Angreifer bestimmt sind (Punkt 4 in der Grafik).

3.11 Weitere Angriffe

Bei einem der getesteten smarten Schlösser in [Rose Ramsey 2016] wird der Befehl zum Öffnen des Passwortes zudem auch für das Setzen eines neuen Passwortes verwendet, indem das aktuelle Passwort gefolgt vom neuen Passwort gesendet wird. Beim Öffnen des Schlosses sind diese beiden Passwörter gleich, beim Ändern ist das zweite Passwort ein anderes. In der Abbildung 11 ist dieser Befehl dargestellt.

Da die Kommunikation unverschlüsselt erfolgt, kann ein Angreifer ohne Weiteres das Passwort des Schlosses ändern, sofern er zuvor das aktuelle Passwort mitgeschnitten hat. Dadurch verliert der Benutzer die Kontrolle über sein Schloss. Um dieses zurückzusetzen, muss die Batterie aus dem Schloss genommen werden, was jedoch nur möglich ist, wenn das Schloss geöffnet ist. Ein Angreifer ist hier also in der Lage den Benutzer sowohl auf Softwareseite, als auch auf Hardwareseite von seinem Schloss auszuschließen [Rose Ramsey 2016].

Des Weiteren war es [Fouladi Ghanoun 2013] möglich ein smartes Schloss, das das Proto-

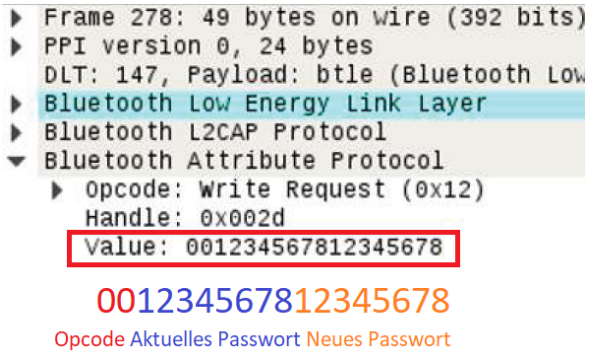

Abbildung 11: Passwortbefehl bei BLE-Schloss (aus [Rose Ramsey 2016])

koll Z-Wave verwendet, zu übernehmen. Beim Verbinden des Schlosses mit einem Z-Wave Controller, über den der Benutzer das Schloss dann steuern kann, wird der Schlüssel für die eingesetzte Verschlüsselung ausgetauscht. Standardmäßig wird ein Zufallswert auf dem Z-Wave Chip generiert, der mit einem fest einprogrammierten 16 Byte langen temporären Standardschlüssel verschlüsselt übertragen wird. Nach dem Schlüsselaustausch werden die Nachrichten mit dem übertragenen Schlüssel verschlüsselt.

Durch einen Implementierungsfehler im Türschloss war es jedoch möglich, dass ein Angreifer den Schlüssel mit einem eigenen überschreibt und somit das Türschloss steuern kann. Im Detail wurde beim Empfang eines Pakets, das den Schlüsselaustausch initiiert, nicht geprüft, ob bereits ein Schlüssel ausgetauscht wurde und nun zur Verschlüsselung verwendet wird. Somit wird bei dieser Initiierung des Schlüsselaustausches der temporäre Schlüssel verwendet, um die Nachrichten beim Schlüsselaustausch zu entschlüsseln. Wenn dieser erfolgreich beendet ist, wird der neu ausgehandelte Schlüssel für die Kommunikation im Folgenden verwendet. Der zuvor verwendete Schlüssel wird überschrieben.

Somit hat der legitime Benutzer dann keine Möglichkeit mehr das Schloss zu steuern, während der Angreifer diese Möglichkeit erhält, da er im Besitz des neuen Schlüssels ist. Die Schwachstelle bestand also darin, dass nicht geprüft wurde, ob bereits ein Schlüssel für die Verschlüsselung der Nachrichten vorhanden ist und wenn, dann wird dieser einfach überschrieben, wenn ein neuer Schlüsselaustausch-Prozess initiiert wird.

Die Arbeit von [Deogirikar Vidhate 2017] sowie das OWASP IoT-Projekt [OWASP 2018] stellen weitere prinzipiell mögliche Angriffe im Bereich des Internet of Things (IoT) vor, zu dem auch die Heimautomatisierung mit ihren Sensoren und Aktoren zählt. Allerdings gibt es zu diesen Angriffen keine Beispiele zu konkreten Implementierungen oder Durchführungen in der Vergangenheit. Hier sind weitere mögliche Angriffe, die für die Heimautomatisierung ebenfalls in Betracht kommen können, wie das Einspielen veralteter Firmware-Updates (Downgrades) erwähnt.

Weitere Beispiele sind aber auch das Social Engineering, bei dem sensible Informationen durch das Agieren mit Personen, die legitimen Zugang zum Netzwerk besitzen, erlangt werden können. Außerdem könnten physikalische Beschädigungen hervorgerufen oder Manipulationen an Geräten durchgeführt werden sowie Schadsoftware (z.B. Viren oder Trojaner) eingesetzt werden, um das System zu schädigen. Da der Fokus jedoch auf den Funkprotokollen der Heimautomatisierung liegen soll, werden diese Art von Angriffen nicht näher betrachtet.

4 Vorgehensweise und Sicherheitsanalyse

Um eine Infrastruktur der Heimautomatisierung strukturiert und systematisch auf Schwachstellen zu untersuchen, die zum einen auf das Protokoll an sich und zum anderen auf Implementierungsfehler der vorliegenden Produkte zurückzuführen sein können, muss zunächst eine Vorgehensweise erarbeitet werden. Mit dieser kann dann die Sicherheitsanalyse durchgeführt werden. Ziel sollte es immer sein die entsprechenden Funktionalitäten eines Geräts steuern zu können, also z.B. Schlösser zu öffnen oder Lampen an- und auszuschalten. Um dies zu erreichen, können die folgenden Punkte systematisch abgearbeitet werden, die aus den zuvor beschriebenen Angriffen resultieren. Da die Vorgehensweise möglichst allgemein gehalten ist, kann es sein, dass einzelne Punkte bei bestimmten Protokollen oder Geräten nicht anwendbar oder in einem speziellen Zusammenhang nicht sinnvoll sind.

4.1 Allgemeine Informationen

Der Fokus dieser Arbeit liegt auf dem Einsatz von Geräten der Heimautomatisierung, die drahtlos kommunizieren. Zunächst einmal gilt es festzustellen, welche Geräte miteinander kommunizieren und erste Informationen über das Netzwerk zu erhalten. Genauer geht es um die Frage, welches Funkprotokoll verwendet wird und um was für eine Art von Geräten es sich handelt. Je nach Gerätetyp könnten spezielle Protokollformen (Profile) im Einsatz sein, wie beispielsweise bei ZigBee als Protokoll bei Beleuchtungskomponenten ZigBee Light Link (ZLL). Je nach Protokoll und Protokollversion können - neben Implementierungsfehlern der Produkte - protokollspezifische Schwachstellen in Frage kommen, die Angriffe ermöglichen können.

Weitere Informationen zum Netzwerk können durch Mithören der Kommunikation erlangt werden. Da es sich um drahtlose Kommunikation mit der Luft als Übertragungsmedium handelt, können die Nachrichten der Geräte im näheren Umfeld mit entsprechender Hardware und Software mitgeschnitten werden. Dabei kommt es ebenfalls auf das eingesetzte Protokoll an, das bestimmte festgelegte Frequenzbänder verwendet und ggf. verschiedene Kanäle auf diesen nutzt.

Durch Mithören der Kommunikation können bereits erste Erkenntnisse gewonnen werden. Diese beinhalten Adressen von Sender und Empfänger von Nachrichten sowie eine Einsicht in den Inhalt der Nachrichten. Daraus kann die Information erhalten werden, ob Nachrichten verschlüsselt sind oder diese im Klartext versendet werden. Beim fehlenden Einsatz von Verschlüsselung ist an dieser Stelle bereits der Erhalt von weiteren ggf. sensiblen Daten wie Passwörter, die beispielsweise zum Öffnen eines Schlosses benötigt werden, oder genaue Befehle zum Steuern von Geräten möglich.

Geräte könnten in bestimmten Intervallen nicht erreichbar sein, da sie für eine bestimmte Zeit in einem sogenannten Sleep-Modus bleiben, um Energie zu sparen. Diese Intervalle könnten durch Mitschneiden der Kommunikation ebenfalls erkennbar werden, z.B. wenn das Gerät Informationen nach dem Zurückkommen aus dem Sleep-Modus beim Koordinator abfragt.

Sofern das eingesetzte Protokoll die Möglichkeit vorsieht, können Informationen über das Netzwerk auch durch den Empfang von sogenannten Beacon-Frames erlangt werden, die in einem festgelegten Intervall ausgesendet werden, um das Netzwerk für außenstehende Geräte sichtbar zu machen, damit diese sich verbinden können. Das Aussenden von Beacon-Frames kann je nach eingesetztem Protokoll auch aktiv ausgelöst werden, beispielsweise bei ZigBee durch Senden eines Beacon Requests. Somit ergibt sich durch Beacon-Frames ebenfalls die Möglichkeit Informationen über ein Netzwerk in der Umgebung zu erhalten.

Zusammengefasst sind also folgende Fragestellungen relevant, wenn es um den Erhalt von ersten Informationen über ein Netzwerk geht:

- Um was für ein Funkprotokoll und ggf. Profil handelt es sich?
- Welche Adressen werden verwendet, was für Geräte gibt es im Netzwerk?
- Wird Verschlüsselung eingesetzt oder werden sensible Daten im Klartext übertragen (z.B. PINs oder Passwörter)?
- Wie sehen Befehle für bestimmte Funktionen aus?
- Gibt es bestimmte Intervalle, bei denen Geräte aus einem Sleep-Modus zurückkehren?
- Gibt es weitere Informationen durch regelmäßige Beacons (Informationsnachrichten)?
- Existieren für das Funkprotokoll Befehle, um explizit Informationen über das Netzwerk zu erfragen (z.B. Beacon Request bei ZigBee)?

4.2 Vertraulichkeit

Um Vertraulichkeit zu erreichen und eben nicht Passwörter oder andere sensible Daten im Klartext zu verschicken, werden häufig Verschlüsselungsverfahren eingesetzt. In den gängigen Standards für Funkprotokolle in der Heimautomatisierung ist der Einsatz von Verschlüsselung vorgesehen, jedoch nicht immer zwingend für die Hersteller von Geräten in diesem Bereich.

Im Folgenden wird zwischen der erstmaligen Verschlüsselung, die den Prozess des Schlüsselaustausches bzw. der Schlüsselverteilung beinhaltet, und dem darauffolgenden Einsatz der Verschlüsselung unterschieden.

4.2.1 Initiierung der Verschlüsselung

Damit allen Partnern der Kommunikation die Schlüssel bekannt sind, mit denen die Nachrichteninhalte verschlüsselt sind, müssen diese zuvor allen Beteiligten mitgeteilt werden. Dies geschieht häufig, indem alle Geräte bestimmte Schlüssel bereits fest einprogrammiert haben, mit denen der initiale Schlüsselaustausch verschlüsselt wird oder indem der Benutzer zunächst einen Code (Passwort/PIN), der von einem Gerät generiert wurde, in ein anderes Gerät eingibt, womit dann Schlüssel berechnet werden.

In ZigBee wird einem neu in das Netzwerk beitretenden Gerät der Netzwerkschlüssel durch das Trust Center mitgeteilt. Im Fall der Verwendung der „Standard“-Security kann dies auch ungesichert geschehen. Durch Mithören der Mitteilung dieses Schlüssels durch das Trust Center - oder im Allgemeinen durch Mithören des Schlüsselaustauschs - beim Eintritt eines Geräts in das Netzwerk könnten Angreifer an den verwendeten Schlüssel gelangen und im Folgenden die Kommunikation entschlüsseln und somit mitlesen.

Schlüssel, die bekannt werden, stellen ebenfalls eine Schwachstelle dar. Wie bereits zuvor beschrieben, wird für ZigBee Light Link beispielsweise ein Masterschlüssel verwendet, um den initialen Schlüsselaustausch zu verschlüsseln. Dieser ist jedoch inzwischen veröffentlicht und somit kann der initiale Schlüsselaustausch mitgelesen werden. Eine ähnlich hohe Gefahr gibt es bei allen fest einprogrammierten Schlüsseln, denn diese können auch bei allen anderen Protokollen früher oder später öffentlich werden. Somit kann es sinnvoll sein, zu prüfen, ob es für das vorliegende Protokoll oder spezielle Geräte veröffentlichte Schlüssel gibt, mit Hilfe derer der Prozess des Schlüsselaustausches entschlüsselt und der eigentliche Schlüssel für die

Kommunikation erhalten werden kann. Fest einprogrammierte Schlüssel, die direkt für die Kommunikation ohne vorhergehenden Schlüsselaustausch verwendet werden, ermöglichen bei öffentlich werden ebenfalls ein Entschlüsseln und Mitlesen der Kommunikation.

Bei der Möglichkeit verschiedene Verfahren zu verwenden, um einen Schlüsselaustausch - oder im Allgemeinen das Verbinden zweier Geräte - durchzuführen, kann es ebenfalls Schwachstellen geben. Nämlich dann, wenn eine Abwärtskompatibilität gewährleistet sein muss und es möglich ist, dass die Geräte durch einen Angreifer dazu gebracht werden können, ein möglichst unsicheres Verfahren zu verwenden, auch wenn Geräte von sich aus ein sicheres Verfahren ermöglichen würden. Dieses macht es einem Angreifer einfacher, an einen Schlüssel zu gelangen. Ein Beispiel dafür ist der Verbindungsprozess bei Z-Wave (unsicherer S0- und sicherer S2-Prozess).

Eine Initiierung des Schlüsselaustausches kann über Netzwerkbefehle erfolgen, wobei ein Koordinator im Netzwerk einen Befehl an das Gerät sendet, das sich mit dem Netzwerk (oder dem Koordinator) verbinden soll. Dabei kann es bei fehlender Überprüfung im Gerät jedoch dazu kommen, dass obwohl das Gerät bereits einen Schlüssel ausgetauscht hat und mit diesem kommuniziert ein erneuter Schlüsselaustausch möglich ist. Dieser könnte durch Angreifer ausgelöst werden. Bei erfolgreichem Abschluss dieses Schlüsselaustausches wird der zuvor gespeicherte Schlüssel mit dem neuen überschrieben, sodass der Angreifer in die Lage versetzt wird verschlüsselte Nachrichten zu senden und somit das Gerät zu übernehmen.

Auch durch Man-in-the-Middle (MITM) könnte ein Schlüssel erlangt werden, indem den beiden Kommunikationspartnern vorgetäuscht wird, dass sie mit dem jeweils anderen kommunizieren.

Fragestellungen bei der Initiierung der Verschlüsselung sind somit:

- Sind Schlüssel fest einprogrammiert?
- Werden Schlüssel unverschlüsselt übertragen, wenn ein neues Gerät beitritt?
- Wird ein temporärer Schlüssel verwendet, um den Schlüsselaustausch zu sichern?
- Sind temporäre Schlüssel für das Funkprotokoll bereits veröffentlicht worden?
- Gibt es Manipulationsmöglichkeiten, sodass ein unsicheres Verfahren verwendet wird (Downgrade)?
- Ist es möglich sich zwischen die Kommunikation zweier Geräte zu schalten (Man-in-the-Middle)?

4.2.2 Verschlüsselung

Sofern es sich bei dem Verfahren zur Verschlüsselung um ein veraltetes, unsicheres oder eigenes Verfahren eines Herstellers handelt, könnten auch Angriffe bekannt sein, mit denen die Verschlüsselung kompromittiert werden kann. Gegebenenfalls werden auch Verfahren eingesetzt, die zwar den Anschein einer Verschlüsselung erwecken, jedoch keinen wirklichen Schutz bieten. Das Umkehren der Reihenfolge von Passwörtern oder PINs wie es bei Bluetooth Low Energy Schlössern bereits vorkam, ist ein solcher Fall. Es gilt also herauszufinden, ob und was für eine Verschlüsselung eingesetzt wird und ob diese den Standards entsprechend in den Geräten umgesetzt wurde.

Des Weiteren kann herausgefunden werden, ob es zusätzlichen Schutz z.B. durch Nonces oder Zeitstempel gibt, die die Verschlüsselung beeinflussen. Somit könnten die gleichen Befehle jedes mal eine andere Verschlüsselung erhalten und es ist für Angreifer nicht immer

gleich ersichtlich, dass es sich um die gleichen Befehle handelt. Anhand der Länge der Verschlüsselung oder je nach Verschlüsselungsverfahren anhand des Aussehens der Zeichenketten (z.B. bestimmte Zeichen sehr oft, andere nur sehr selten) bei bestimmten Befehlen, könnten Schlussfolgerungen über den verschlüsselten Befehl möglich sein. Beispielsweise könnte ein Befehl zum Öffnen eines Schlosses eine sehr lange Zeichenkette nach der Verschlüsselung bewirken und der Befehl zum Schließen eine kurze Zeichenkette. Daraus könnte ein Angreifer unter Umständen also ableiten um welchen Befehl es sich handelt, ohne den Inhalt der verschlüsselten Nachricht zu kennen.

Des Weiteren können Schlüssel, die für das Entschlüsseln von Nachrichten verwendet werden, öffentlich werden. Dies kann der Fall sein, wenn Netzwerkschlüssel oder vergleichbare Schlüssel standardmäßig in der Firmware der Geräte fest einprogrammiert sind und vom Anwender nicht geändert wurden oder es keine Möglichkeit gibt diese zu ändern.

Daraus ergeben sich die für diesen Punkt relevante Fragestellungen:

- Gibt es wirklich eine Verschlüsselung oder wird nur der Anschein einer Verschlüsselung geweckt (z.B. Umkehren von Zeichen-Reihenfolgen)?
- Wird ein sicheres Verfahren zur Verschlüsselung verwendet?
- Werden gleiche Befehle immer gleich verschlüsselt?
- Werden bereits bekannte Schlüssel verwendet?
- Sind für das verwendete Verschlüsselungsverfahren Sicherheitslücken bekannt?
- Können Rückschlüsse auf den Inhalt anhand von Nachrichtenlängen oder verwendeten Zeichen gemacht werden?

4.3 Integrität

Die Integrität bezeichnet die Korrektheit von Daten, insbesondere auch von Befehlen und Nachrichten im Allgemeinen. Sie sollen von Angreifern nicht unbemerkt verändert werden können. Wird jedoch keine geeignete Maßnahme ergriffen, um Manipulationen zu verhindern, so können Befehle verändert oder auch eigene erstellt werden. Beim fehlenden Einsatz von Verschlüsselung sind aufgenommene Befehle nicht nur wiedereinspielbar, sondern es ist auch die Erstellung eigener Befehle denkbar.

Im Folgenden wird zwischen dem Verändern von Befehlen und fehlerhaften oder unvollständigen Integritätsprüfungen unterschieden.

4.3.1 Veränderte Befehle

Je nach Protokoll und der Verwendung von Sicherheitsfunktionen kann es für Angreifer möglich sein, zuvor aufgenommene Nachrichten zu verändern oder auch gänzlich neue Nachrichten zu erstellen. Besonders bei fehlenden oder nicht korrekt implementierten Integritätsprüfungen oder nicht vorhandener oder unsicherer Verschlüsselung kann es bereits ausreichen Nachrichten aufzuzeichnen, diese zu verändern und einzuspielen.

Bei gängigen - nicht proprietären - Protokollen können Aufbau und erforderliche Inhalte der Befehle in der Regel aus den jeweiligen Spezifikationen entnommen werden. Der Aufbau der Befehle kann auch durch eine Analyse zuvor aufgenommener Nachrichten aus dem Punkt 4.1 bekannt sein. Dafür könnte es unter Umständen hilfreich sein, Software zu dekompilieren - sofern dies möglich ist -, um entsprechende Stellen im Code zu analysieren und dadurch

herauszufinden wie Befehle aufgebaut sind. Sofern Verschlüsselung eingesetzt wird, ist für das Erstellen von eigenen Befehlen ein gültiger Schlüssel erforderlich, der ggf. aus dem Punkt 4.2.1 erhalten wurde.

Durch sogenanntes Fuzzing - zufälliges Verändern von Nachrichten -, was auch bei verschlüsselten Nachrichten ohne gültigen Schlüssel möglich ist, konnten in der Vergangenheit bereits Geräte in einen Fehlerstatus gebracht werden, wodurch es zu unvorhersehbarem Verhalten der Geräte kam. Bei bestimmten Implementierungen kann ein solcher Fehlerstatus beispielsweise dazu führen, dass sich ein Schloss öffnet und ungewollt Zugang gewährt.

Relevant sind in diesem Fall die Fragestellungen:

- Wie sind Befehle aufgebaut (ggf. nach Dokumentation bei standardisierten Funkprotokollen)?
- Können aufgenommene Befehle systematisch verändert werden?
- Können aufgenommene Befehle zufällig verändert werden?
- Können Geräte durch Einspielen fehlerhafter Nachrichten zu Problemen gebracht werden (z.B. Fehlerstatus)?

4.3.2 Integritätsprüfungen

Sofern bei Nachrichten Integritätsprüfungen vorgesehen sind, könnten diese vor unentdeckten Manipulationen schützen. Es kann jedoch vorkommen, dass diese Prüfungen lediglich darauf abzielen vor Bitübertragungsfehlern zu schützen und manipulierbar sind oder dass die Prüfwerte nicht miteinander verglichen werden.

Eine nicht vorhandene Integritätsprüfung könnte dazu führen, dass Nachrichten einfach manipulierbar sind. Zudem könnte die Prüfung nicht ordnungsgemäß implementiert sein und Werte der Prüfung beim Endgerät nicht miteinander verglichen werden, was jedoch notwendig ist, um Veränderungen an den Nachrichten zu erkennen.

Somit sind folgenden Fragestellungen in diesem Zusammenhang relevant:

- Wird eine Integritätsprüfung durchgeführt?
- Worauf zielt die Integritätsprüfung ab (nur Entdeckung von Übertragungsfehlern oder auch aktive Manipulation)?
- Werden Prüfsummen wirklich geprüft (beim Empfänger selbst berechnet und miteinander verglichen)?
- Kann die Integritätsprüfung manipuliert werden?

4.4 Verfügbarkeit

Die Verfügbarkeit bei der Heimautomatisierung besagt, dass ein Anwender seine Geräte möglichst zeitnah steuern können soll. Es existieren zwei verschiedene Möglichkeiten diese Verfügbarkeit einzuschränken und es dem Benutzer schwierig oder unmöglich zu machen seine Geräte zu steuern: Zum einen kann der Funkkanal blockiert werden und zum anderen können einzelne Geräte nicht mehr erreichbar werden.

4.4.1 Blockieren des Funkkanals

Damit Geräte für den Benutzer nicht mehr erreichbar sind, kann die Kommunikation zwischen Zentrale und dem zu steuernden Gerät unterbunden werden. Dies kann realisiert werden, indem der Funkkanal durch das Aussenden vieler Nachrichten oder Störsignale blockiert wird. Somit ist es für den Benutzer nicht mehr möglich seine Geräte zeitnah zu steuern, da Nachrichten entweder gar nicht erst versendet werden können, weil der Funkkanal für das Gerät gerade nicht frei ist oder Nachrichten werden mit Störsignalen so beeinträchtigt, dass das Gerät, für den diese bestimmt sind, diese nicht oder nicht korrekt empfangen kann.

Die folgenden Fragen sind für diesen Bereich von Bedeutung:

- Gibt es eine Möglichkeit den Ausfall von Geräten durch Störungen zu verhindern oder auf andere Kanäle auszuweichen (z.B. bei mehreren Funkkanälen)?
- Kann eine große Zahl von Nachrichten die Kommunikation im Netzwerk blockieren?
- Wird der Benutzer informiert, wenn Befehle nicht ausgeführt wurden?
- Werden Befehle automatisch erneut abgesendet, wenn diese nicht ankamen oder bestätigt wurden?
- Wird der Benutzer informiert, wenn Geräte nicht mehr erreichbar sind?
- Wird der Benutzer durch Störsignale dazu gebracht, Geräte neu zu verbinden oder zurückzusetzen (z.B. durch Fehlermeldungen)?

4.4.2 Nicht-Erreichbarkeit von Geräten

Die Verfügbarkeit kann ebenfalls behindert werden, indem einzelne Geräte nicht mehr erreichbar werden. Da ein Großteil der Geräte in der Heimautomatisierung mit Batterien betrieben wird und über keine ständige Stromversorgung verfügt, können diese Geräte auf Grund von mangelnder Energieversorgung ausfallen. Um das Leeren der Batterien zu beschleunigen, können große Anzahlen von Nachrichten an die Geräte gesendet werden, die beliebigen Text beinhalten. Dieser muss dann - auch beim Einsatz von Verschlüsselung und/ oder Integritätsprüfungen - auf dem Gerät verarbeitet werden und sorgt so für einen höheren Energiebedarf. Dies führt zu einem schnelleren Leeren der Batterien und letztendlich zum Ausfallen des Geräts, wenn eine minimale Ladung zum Betreiben des Geräts unterschritten wird. Somit ist das Gerät für den Benutzer nicht mehr erreichbar.

Bei ZigBee-Netzwerken kann diese Vorgehensweise mit dem Senden vieler Nachrichten geschehen, die verarbeitet werden müssen. Insbesondere durch die kontinuierliche Berechnung von Integritätsprüfungen oder Verschlüsselung kann dies zu hohem Energiebedarf führen. In Bluetooth Low Energy kann das Erhöhen des Energiebedarfs mit einer großen Zahl von Schreibzugriffen oder Verbindungsanfragen geschehen. Schwachstellen können zudem beim Neustart des Geräts nach dem Laden oder Wechseln der Batterie auftreten, indem z.B. flüchtige Variablen zurückgesetzt werden.

Die Fragestellungen für diesen Fall wären die Punkte:

- Mit welcher Stromquelle wird das Gerät betrieben (z.B. Batterie/Akku oder Strom per Kabel)?
- Welche Operationen benötigen eine größere Menge Energie (z.B. Schreibzugriffe, Berechnungen für Integritätsprüfungen)?
- Wie erfährt der Benutzer von einem Ausfall eines Geräts?

- Was passiert nach Neustart des Geräts (z.B. Zurücksetzen von Countern, Schlüsselwechsel)?
- Hat ein Geräteausfall weitere Konsequenzen?

4.5 Authentizität

Mit der Authentizität soll sichergestellt werden, dass z.B. Nachrichten von einem legitimen Absender stammen und die Identität, die dieser angibt, korrekt ist. Um dies zu gewährleisten, können zum einen Maßnahmen auf Softwareebene getroffen werden, wie beispielsweise mit Message Authentication Codes (MAC), aber zum anderen auch auf physikalischer Ebene, wie das Zulassen von Nachrichten, wenn Sender sich in einem geringen Radius um ein Gerät befinden.

4.5.1 Benutzerauthentifizierung

Je nach Infrastruktur könnte es möglich sein, dass bestimmte Aktionen nur von bestimmten Benutzern ausführbar sind. Geräte, die nicht im Netzwerk sind, oder Geräte von Angreifern generell sollten Aktionen nicht ausführen können. Es könnte also hilfreich sein, wenn sich legitime Benutzer als diese ausweisen können. Ein legitimer Benutzer könnte also sein Passwort eingeben, um bestimmte Funktionen ausführen zu können.

Eine solche Authentifizierung könnte jedoch auch über den Austausch von bestimmten Informationen geschehen, die als „Geheimnisse“ zwischen legitimem Benutzer und dem Gerät fungieren. Es wird davon ausgegangen, dass ein Angreifer nicht im Besitz dieser Informationen ist. Dies kann beispielsweise ein Passwort sein oder ein Schlüssel, der zunächst beim Benutzer vorhanden sein muss und mit denen sich der Benutzer als legitimer Benutzer zu erkennen geben kann.

Dabei können die folgenden Fragestellungen relevant sein:

- Gibt es eine Benutzerauthentifizierung?
- Wie werden Benutzer authentifiziert (z.B. Benutzernamen, Passwörter, MAC-Adresse, ausgetauschte „Geheimnisse“ wie Schlüssel)?
- Sind bestimmte Befehle nur für bestimmte Benutzer möglich?

4.5.2 Authentizität durch Nähe zum Gerät

Bei bestimmten Protokollen wie ZigBee können bestimmte Befehle ausgeführt werden, die unabhängig vom Netzwerk des Geräts sind. Sie dienen zum Steuern des Geräts z.B. bevor es einem Netzwerk beitritt, indem sie das Zurücksetzen oder Eintreten in ein Netzwerk ermöglichen sollen. Damit nicht jedes beliebige Gerät diese Befehle absenden und ausführen kann, ist im Standard vorgesehen, dass die Entfernung des Senders zum Gerät überprüft werden soll und beim Überschreiten einer bestimmten Entfernung sollen die Befehle verworfen und somit nicht ausgeführt werden. So soll ein legitimer Besitzer authentifiziert werden, da nur er sich in einem so geringen Bereich (Abstand meist unter einem Meter) aufhalten können soll.

Durch Fehler in der Implementierung der Befehle oder der Prüfung der Entfernung des Senders kann es jedoch möglich sein, dass diese Befehle auch von weiteren Entfernungen aus ausgeführt werden können. Die Fragestellung dabei ist, ob die Entfernungsprüfung auch

wirklich bei jedem Befehl durchgeführt wird oder nur bei einzelnen. Dies führt dazu, dass beliebige Geräte, auch in weiterer Entfernung als der festgelegten, in die Lage versetzt werden können Befehle auszuführen, zu denen sie eigentlich nicht befähigt sein sollten. Insbesondere ein Befehl zum Zurücksetzen des Geräts kann eine Übernahme des Geräts in ein Netzwerk eines Angreifers erlauben.

Relevante Fragestellungen in diesem Zusammenhang sind:

- Gibt es für das Funkprotokoll Befehle, die unabhängig vom Netzwerk abgesendet werden können?
- Gibt es Befehle, die eine physikalische Nähe zum Gerät voraussetzen?
- Wie wird diese Entfernung überprüft?
- Wird für alle diese Befehle die Entfernung überprüft?
- Können Geräte unautorisiert zurückgesetzt werden?

4.5.3 Firmware-Updates

Einige Funkprotokolle bieten die Möglichkeit Firmware-Updates einzuspielen, damit Hersteller beispielsweise Sicherheitslücken in vergangenen Versionen schließen können. Diese Updates sollten authentifiziert sein, d.h. nur der Hersteller sollte in der Lage sein Updates zu erstellen. Oft ist in den Standards zwar festgelegt wie ein Update auf dem Gerät erhalten werden soll, jedoch nicht genauer spezifiziert wie ein solches Update gesichert sein soll. Hersteller verwenden hier also Verfahren nach ihrem eigenen Ermessen.

So kann es sein, dass Updates entweder unverschlüsselt sind oder nur ungenügend gesichert sind. Bei unverschlüsselten Updates besteht die Möglichkeit ggf. fest einprogrammierte Schlüssel auszulesen, die ein Mitlesen der Kommunikation oder des Schlüsselaustausches ermöglichen. Die Verwendung eines symmetrischen Verschlüsselungsverfahrens könnte dafür sorgen, dass bei öffentlich werden des Schlüssels Angreifer ebenfalls Updates erstellen bzw. vorhandene verändern können. Wenn die Verschlüsselungsverfahren nicht festgelegt sind, könnten auch potentiell unsichere Verfahren verwendet werden, die leicht zu kompromittieren sind.

Außerdem kann eine weitere Sicherheitslücke beim Einspielen von Updates sein, dass Geräte auch ältere Versionen von Firmware als die aktuell installierte Version annehmen, also Downgrades erlauben. Dadurch könnten ältere Firmware-Versionen, die weitere Sicherheitslücken aufweisen, von Angreifern eingespielt werden und diese Sicherheitslücken dann ausgenutzt werden.

In diesem Punkt bedeutsame Fragen sind:

- Gibt es die Möglichkeit von Firmware-Updates?
- Werden Updates verschlüsselt oder unverschlüsselt übertragen?
- Sind Updates symmetrisch oder unsymmetrisch verschlüsselt?
- Sind Schlüssel für Updates bereits bekannt?
- Können in Updates fest einprogrammierte Schlüssel ausgelesen werden?
- Sind Downgrades (Einspielen älterer Updates) möglich?

4.6 Nicht-Vermehrbarkeit

Befehle und Nachrichten im Allgemeinen sollten von Angreifern nicht kopiert werden können. Dafür werden häufig Vorkehrungen getroffen wie das Verwenden von Verschlüsselung mit Zeitstempeln oder Zählern, die mit jedem Befehl hochgezählt werden. Damit soll das spätere Wiedereinspielen und Ausführen der Befehle von zuvor aufgenommenen Nachrichten verhindert werden.

4.6.1 Wiedereinspielen von Befehlen

Bei sogenannten Replay Angriffen können Nachrichten aufgenommen und wiedereingespielt werden, sodass die Befehle darin erneut ausgeführt werden. Bei unverschlüsselter Kommunikation ist das Einspielen mit wenig Aufwand möglich. Sofern ein Counter verwendet wird, der mit dem Senden jeder Nachricht erhöht wird, kann dieser manipuliert werden, wenn er ebenfalls unverschlüsselt ist. Er muss beim Einspielen der Nachricht höher sein, als der Zähler der zuletzt gesendeten Nachricht.

Zudem muss ein Counter (oder auch andere Vorkehrungen gegen Wiedereinspielen, z.B. Zeitstempel oder Nonces) beim Empfänger geprüft werden. Das bedeutet es muss der empfangene Counter-Wert mit dem zuletzt empfangenen Wert, der in der Regel flüchtig gespeichert wird, verglichen werden und Pakete, die einen niedrigeren Counter besitzen, müssen verworfen werden. Eine analoge Vorgehensweise gilt bei veralteten Zeitstempeln und bereits verwendeten Nonces.

Bei verschlüsselten Nachrichten kann es ebenfalls möglich sein Replay Angriffe erfolgreich durchzuführen. Dies ist zum Beispiel der Fall, wenn keine Vorkehrungen wie Counter oder Nonces getroffen wurden oder diese nicht ordnungsgemäß beim Empfang der Nachricht überprüft werden sowie bei nicht stattfindenden Schlüsselwechseln. Der Angreifer muss dafür nicht im Besitz eines gültigen Schlüssels sein, da die Nachricht zuvor bereits aufgenommen wurde und so vom Empfänger entschlüsselt werden kann.

Auch nach Neustarts von Geräten, wie sie beispielsweise durch Energy Depletion Angriffe ausgelöst werden können, wenn das Gerät wieder eine Stromversorgung bekommt, können Replay Angriffe möglich werden. Temporäre Werte wie Counter werden nicht permanent gespeichert und bei Neustarts werden in der Regel diese Werte zurückgesetzt. Somit wäre es also möglich nach dem Neustart Nachrichten mit Countern wieder einzuspielen, da der Counter-Wert im Gerät dann bei *0* liegt.

Ähnlich kann es sich verhalten, wenn die maximalen Werte der Counter-Variablen erreicht werden. Counter haben in der Regel eine feste Größe zugewiesen und wenn diese überschritten wird, beginnt die Zählung wieder bei *0* und somit können ebenfalls Replay-Angriffe möglich werden.

Relevant für diesem Fall sind die Fragestellungen:

- Werden für die gleichen Aktionen immer die gleichen Befehle gesendet oder unterscheiden diese sich?
- Ist ein Einspielen von aufgenommenen Nachrichten möglich?
- Gibt es Vorkehrungen gegen Replay-Angriffe (z.B. Counter, Zeitstempel, Nonces)?
- Können diese Vorkehrungen manipuliert werden?
- Werden diese Vorkehrungen korrekt umgesetzt (z.B. werden Counter verglichen)?
- Gibt es regelmäßige Schlüsselwechsel?

- Werden Counter zu bestimmten Zeitpunkten wieder zurückgesetzt (z.B. Neustarts, Erreichen eines maximalen Wertes)?

4.6.2 Klonen/Fälschen von Geräten

Je nach Protokoll und implementierter Möglichkeiten zur eindeutigen Identifizierung und Authentifizierung von Geräten, kann es möglich sein Geräte zu klonen oder zu fälschen. Ein Angreifer könnte sein Gerät nach außen hin so aussehen lassen, als wäre es ein legitimes Gerät im Netzwerk und so an Daten kommen, die nicht für ihn bestimmt sind, eigene Daten senden oder Daten manipulieren.

Die Manipulation von MAC-Adressen oder weiteren protokollspezifischen Adressen kann unter Umständen bereits ausreichen, um einem Gerät vorzutäuschen, es handele sich um ein anderes bekanntes Gerät. Durch die übereinstimmende Einrichtung von Services und Characteristics bei BLE oder gleichen Profilen bei ZigBee können Geräte dazu gebracht werden, ein geklontes Gerät für ein legitimes Gerät zu halten und Daten mit diesem auszutauschen. Es kann dadurch zum Erlangen von Daten durch den Angreifer oder aber zu Problemen mit der Verfügbarkeit aus Sicht des Benutzers kommen, da dieser nicht mehr das legitime Gerät steuert und dieses entsprechend auch nicht auf Befehle reagiert.

Die folgenden Fragestellungen sind hier von Bedeutung:

- Anhand welcher Merkmale werden Geräte identifiziert (z.B. MAC-Adresse)?
- Können diese Merkmale manipuliert/verändert werden?
- Können mit gefälschten Merkmalen Daten erlangt werden?
- Wird der Benutzer informiert, wenn mehrere Geräte die gleichen Merkmale aufweisen?
- Gibt es sichere Maßnahmen zur Authentifikation von Geräten?

5 Vorbereitungen zur Sicherheitsanalyse

Mit Hilfe der zuvor vorgestellten Vorgehensweise und den entsprechenden Fragestellungen sollen nun verschiedene Geräte eines Protokolls analysiert und deren Sicherheit in Bezug auf die festgelegten Sicherheitsziele untersucht werden. Dafür wird zunächst ein Protokoll festgelegt, das untersucht werden soll. Im Anschluss werden verschiedene Geräte ausgewählt, die dieses Protokoll implementieren sowie die Analyse durchgeführt.

5.1 Auswahl Funkprotokoll

Die bekanntesten und am meisten verbreiteten Funkprotokolle im Bereich der Heimautomatisierung sind die zu Beginn der Arbeit vorgestellten Protokolle ZigBee, Bluetooth (Low Energy) und Z-Wave. Weitere - unter anderem proprietäre Protokolle - sind ebenfalls existent. Ein einheitlicher Standard für alle Geräte der Heimautomatisierung existiert nicht.

Für die Auswahl der zu analysierenden Geräte ist es sinnvoll, Geräte zu wählen, die das gleiche Funkprotokoll verwenden, um eine gute Vergleichbarkeit zu erhalten und nicht alle Vorgänge für unterschiedliche Protokolle implementieren zu müssen. Zudem würden unterschiedliche Protokolle unter Umständen auch unterschiedliche Hard- und Softwarekomponenten erforderlich machen, sodass hier ein deutlich höherer Ressourcenbedarf besteht und der Aufwand insgesamt deutlich ansteigt. Aus diesen Gründen wird eines der Funkprotokolle ausgewählt und dann verschiedene Geräte ausgesucht, die dieses Funkprotokoll verwenden, an denen die Sicherheitsanalyse durchgeführt wird.

Geeignet für eine Sicherheitsanalyse erscheinen vor allem die am meisten verbreiteten Funkprotokolle ZigBee, BLE und Z-Wave, da diese bereits häufig in Haushalten vorzufinden sind. Durch die große Verbreitung und Bekanntheit könnten Anwender annehmen, dass diese Protokolle und die Produkte, die diese verwenden, auch entsprechend sicher sind. Sicher bedeutet in diesem Fall, dass sie ausgestattet sind mit Sicherheitsvorkehrungen vom Stand der Technik, optimalerweise ohne Schwachstellen und optimalerweise ohne Möglichkeiten Angriffe von außen durchzuführen.

Die Wahl des Funkprotokolls, das analysiert werden soll, fiel auf Bluetooth Low Energy (BLE), da es eines der bekanntesten Funkprotokolle in der Heimautomatisierung ist. Da bereits Bluetooth eine sehr hohe Verbreitung hat - und in den meisten mobilen Geräten enthalten ist - besitzt auch Bluetooth Low Energy ein großes Potential in Zukunft eine große Bekanntheit zu erlangen und auf Grund seines geringen Energiebedarfs vor allem im Bereich der Heimautomatisierung Verwendung zu finden. Zudem ist BLE nicht proprietär.

Der Vorteil für die Analyse besteht unter anderem darin, dass BLE-Geräte in der Regel nicht über eine zusätzliche Zentrale gesteuert werden müssen, die die Befehle eines Benutzers in entsprechende Befehle des jeweiligen Protokolls umwandelt. Neuere mobile Endgeräte, die die Möglichkeit zu Bluetooth 4.0 oder höher besitzen, sind in der Lage direkt per Bluetooth Low Energy mit den jeweiligen Geräten zu kommunizieren. Ein Nachteil, der dadurch entsteht - zwar nicht für die Sicherheitsanalyse, sondern eher für den Endbenutzer -, ist die Entfernung des Benutzers zu seinem Gerät. Dieser muss im Empfangsbereich von Bluetooth Low Energy bleiben, um das Gerät steuern zu können und eine Kommunikation über das Internet (z.B. wenn der Benutzer sein Gerät von unterwegs steuern möchte) ist somit nicht möglich. Dafür würde wieder eine Zentrale benötigt, die die Befehle aus dem Internet entgegennehmen und in BLE-Befehle umwandeln kann.

Typische Geräte in der Heimautomatisierung, die mit Bluetooth Low Energy als Funkprotokoll arbeiten, sind z.B. Lampen oder Schlösser, wie sie bereits in zuvor vorgestellten Angriffen erwähnt wurden.

5.2 Auswahl von Hard- und Software zur Analyse

Es wird außerdem eine Auswahl an Hard- und Software getroffen, die notwendig ist, um die Sicherheitsanalyse durchzuführen. Es handelt es sich dabei um Geräte, die zum Senden und Empfangen von Bluetooth-Befehlen benötigt werden. Diese bieten in Kombination mit geeigneter Software die Möglichkeit, die relevanten Fragestellungen in den einzelnen Kategorien der Vorgehensweise beantworten zu können.

Es wird daher im Folgenden zwischen den Punkten Hardware sowie Software zur Analyse unterschieden. Die Hardware, die analysiert werden soll - also Geräte, die untersucht werden - wird in den jeweiligen Unterpunkten der Sicherheitsanalyse im Kapitel 6 vorgestellt.

5.2.1 Hardware

Da es sich bei der Auswahl des Funkprotokolls um Bluetooth Low Energy handelt, ist bestimmte Hardware notwendig, um Nachrichten in BLE mitzuschneiden und eigene Nachrichten zu senden. Bluetooth und auch Bluetooth Low Energy arbeiten auf dem 2,4-GHz-Frequenzband, bei BLE unterteilt in 40 Kanäle (davon 3 als Advertising-Kanäle). Beide verwenden unter anderem Frequency Hopping - ein schnelles wechseln der Kanäle, um die Störanfälligkeit zu senken -, da beispielsweise WLAN im gleichen Frequenzbereich arbeitet.

Es existieren verschiedene Mikrocontroller, die in der Lage sind Bluetooth-Kommunikation zu empfangen und zu senden. Die Bluetooth Version dieser Chips sollte mindestens bei 4.0 liegen, da diese Bluetooth Low Energy beinhaltet. Damit es möglichst einfach ist, die Funktionalitäten mit einem Rechner zu steuern und die Daten mit diesem auszuwerten und zu verarbeiten, gibt es diese Mikrocontroller bereits als fertige USB-Sticks von verschiedenen Anbietern, sodass diese nur noch in einen USB-fähigen Rechner eingesteckt werden müssen und dann gesteuert werden können.

Als Hardware zum Senden und Empfangen von Bluetooth Low Energy Signalen wird der USB-Stick „CC2540 USB-Dongle“ von Texas Instruments ausgewählt. Nach einer kurzen Installation der zugehörigen Treiber ist das Gerät einsatzbereit und kann verwendet werden, um sämtliche Kommunikation in Bluetooth Low Energy im Umfeld zu empfangen (sogenanntes „sniffen“). Ebenfalls ist ein „BBC micro:bit (MB80)“ vorhanden, mit dem ebenfalls die Kommunikation mitgeschnitten werden kann. Mit diesem sind zudem noch einige weitere Funktionalitäten möglich, die im Abschnitt 5.2.2 beschrieben werden.

Beide Geräte sind in den Abbildungen 12 und 13 dargestellt.

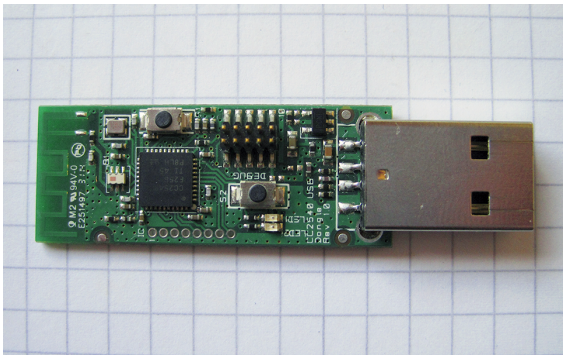

Abbildung 12: Texas Instruments CC2540 USB-Dongle

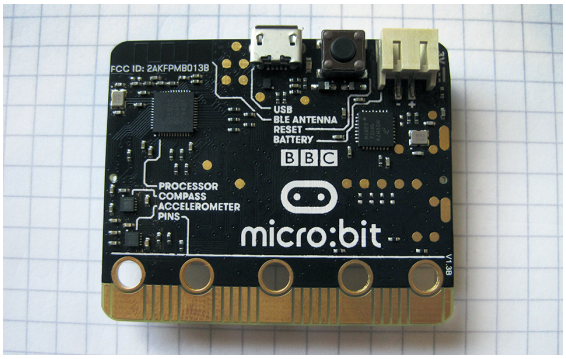

Abbildung 13: BBC micro:bit (MB80)

Zum Senden von eigenen Befehlen per Bluetooth Low Energy wird ein in einen Laptop eingebauter Bluetooth Chip von Intel mit entsprechender Software verwendet.

5.2.2 Software

Texas Instruments liefert für seinen CC2540 USB-Stick bereits eine Software mit dem Namen „Texas Instruments SmartRF Packet Sniffer“ mit, die es ermöglicht die empfangenen Bluetooth Low Energy Daten anzuzeigen. Die Software existiert nur für Windows-Betriebssysteme (ab Windows XP) und es ist nicht möglich eigene Befehle bzw. Nachrichten auszusenden, lediglich das Mitlesen ist möglich. Sie stellt alle aufgenommenen Bluetooth Low Energy Befehle farblich gekennzeichnet dar und ermöglicht auch das Filtern bestimmter Nachrichten nach eingegebenen Merkmalen (z.B. nur Nachrichten einer bestimmten MAC-Adresse). Ein Ausschnitt dieser Software und dessen grafischer Darstellung am Beispiel der Kommunikation mit einem smarten Vorhängeschloss ist in der Abbildung 14 dargestellt.

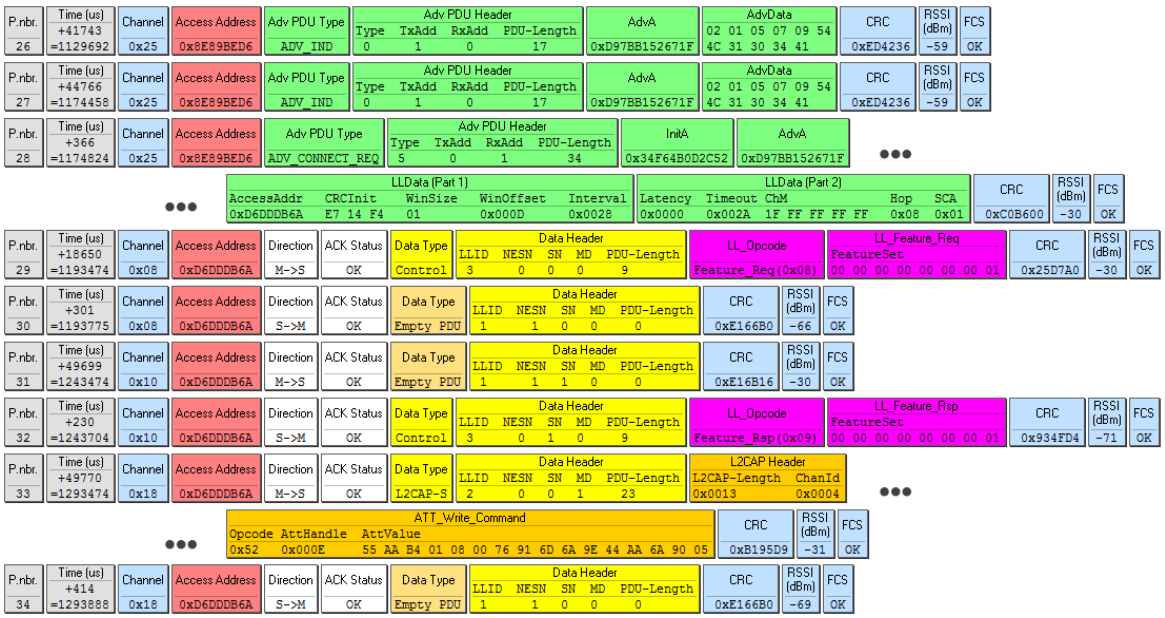

Abbildung 14: Texas Instruments SmartRF Packet Sniffer

Jede Nachricht wird in einer Zeile dargestellt, wobei Advertising-Pakete grün angezeigt werden und - sobald eine Verbindung zu Stande kommt - werden Verbindungspakete gelb dargestellt, wobei Verbindungseinstellungen lila und Nachrichten wie Schreibbefehle orange gefärbt sind. Zudem werden die Bestandteile der Teilnachrichten wie Header oder CRC-Prüfsumme als Überschriften angezeigt und in Kästen aufgeteilt, sodass ein schneller Überblick über die empfangenen Pakete möglich ist. Im Rohformat, in dem die Befehle gesendet werden, handelt es sich dabei lediglich um eine Anreihung von Hexadezimalzeichen, die mit Hilfe der Kenntnisse über den BLE-Standard aufgeteilt und analysiert werden müssten.

Zudem können mit der Software „btlejack“, die zusammen mit dem BBC micro:bit verwendet wird, verschiedene Angriffe durchgeführt werden, wie z.B. das Senden von Störsignalen, sodass Verbindungen unterbrochen werden. Ebenfalls ist es möglich die Kommunikation mitzuschneiden, auch wenn eine Verbindung bereits aufgebaut wurde und die Kommunikation bereits stattfindet. Mit dem CC2540 ist das Mitschneiden nur möglich, wenn die Kommunikation bereits vor dem Verbindungsaufbau aufgenommen wird. Allerdings gibt es in diesem Fall beim Mitschneiden keine grafische Aufbereitung der Daten wie es beim TI Packet Sniffer der Fall ist, sondern es werden lediglich die Nachrichten im Rohformat angezeigt.

Es ist zudem weitere Software für die Sicherheitsanalyse erforderlich, um eigene Befehle und Scripts zu erstellen. Für das Betriebssystem Linux gibt es bereits eine Zahl von Tools und Frameworks, die es ermöglichen BLE-Befehle zu senden und zu empfangen. Beispiele

dafür sind „blueZ“ und „bluepy“. Das erste Tool ermöglicht mit seinem „gatttool“ einige Befehle, die manuell via Konsole abgesendet werden können (z.B. Schreibbefehle oder Verbindungsanfragen). Letzteres ist ein Python-Framework, das auf „blueZ“ aufbaut. Weitere Implementierungen bauen wiederum auf „bluepy“ auf, beispielsweise „BLEAH“. Diese Software ist insbesondere dafür ausgelegt um Bluetooth Low Energy Geräte anzugreifen. Damit ist es z.B. möglich alle BLE-Geräte in der Umgebung aufzulisten und die vorhandenen Characteristics und entsprechende Lese- und Schreibrechte zu erfahren. Ein Beispiel für die Auflistung dieser Rechte (im Fall der im Folgenden vorgestellten smarten Lampe) zeigt die Abbildung 15. Außerdem können Lese- und Schreibbefehle abgesendet werden. „BLEAH“ bietet jedoch nicht die Möglichkeit durchgehend allen Datenverkehr in BLE mitzuschneiden, sodass insgesamt nicht nur eine einzige Software verwendet werden kann, um die Sicherheitsanalyse durchzuführen, sondern mehrere Komponenten zusammenspielen.

```
f8:1d:78:62:e9:23 (-71 dBm)
Vendor                  | Ieee Registration Authority
Allows Connections      | ✓
Address Type            | public
Flags                   | LE Limited Discoverable, BR/EDR Not Supported
Complete 16b Services   | '0000fff0-0000-1000-8000-00805f9b34fb'
Complete Local Name     | LEDBLE-7862E923
Appearance              | '0000'
```

Handles	Service > Characteristics	Properties	Data
0001 -> 0007	Generic Access (00001800-0000-1000-8000-00805f9b34fb)		
0003	Device Name (00002a00-0000-1000-8000-00805f9b34fb)	READ	u'LEDBLE-7862E923\x00'
0005	Appearance (00002a01-0000-1000-8000-00805f9b34fb)	READ	Generic Keyring
0007	Peripheral Preferred Connection Parameters (00002a04-0000-1000-8000-00805f9b34fb)	READ	Connection Interval: 0 -> 0 Slave Latency: 0 Connection Supervision Timeout Multiplier: 0
0009 -> 000b	ffe5 (0000ffe5-0000-1000-8000-00805f9b34fb)		
000b	ffe9 (0000ffe9-0000-1000-8000-00805f9b34fb)	**WRITE NO RESPONSE** **WRITE**	
000c -> ffff	ffe0 (0000ffe0-0000-1000-8000-00805f9b34fb)		
000e	ffe4 (0000ffe4-0000-1000-8000-00805f9b34fb)	NOTIFY	

Abbildung 15: Auflisten der Lese- und Schreibrechte mit BLEAH

In der Abbildung 15 ist im oberen Kasten zunächst eine Übersicht über allgemeine Informationen des Geräts dargestellt, beispielsweise, dass es Verbindungen im Allgemeinen zulässt oder ein Gerätename, in diesem Fall der standardmäßige Name der Lampe. Im unteren Kasten sind Informationen über alle Services und Characteristics aufgeführt. Es gibt es hier verschiedene Attribute, die einen Lesezugriff oder auch Schreibzugriff auf bestimmte Handles ermöglichen. Auf Handle *0x000B* ist z.B. ein Schreibzugriff möglich. So ist es möglich einen ersten Überblick über die Möglichkeiten und Eigenschaften des Geräts zu erhalten.

Außerdem können Shell-Scripte oder auf Grundlage von „bluepy“ auch eigene Python-Scripte erstellt und ausgeführt werden und somit eigene Funktionalitäten (z.B. mehrere Befehle hintereinander) implementiert werden.

Um den gesamten Datenverkehr mitschneiden und analysieren zu können, wird ein Rechner mit Windows 10, der Texas Instruments CC2540 und der SmartRF Packet Sniffer von Texas Instruments verwendet. Um erste Erkenntnisse über die Geräte zu erlangen und einfachere BLE-Befehle zu senden, wird ein Ubuntu Betriebssystem mit „BLEAH“ verwendet, wobei hier auch der im Rechner eingebaute Bluetooth-Chip als Sender fungieren kann, damit parallel dazu der Datenverkehr mit dem CC2540 mitgeschnitten werden kann. Außerdem können eigene Scripte zusätzlich unterstützen, wenn bestimmte Funktionalitäten mit existierenden Tools nicht möglich sind.

5.3 Anpassung der Analyse

Wie bereits zuvor beschrieben, handelt es sich bei der Sicherheitsanalyse um eine möglichst allgemeine Vorgehensweise, die unabhängig von bestimmten Funkprotokollen sein soll. Aus

diesem Grund kann es möglich sein, dass bestimmte darin enthaltene Punkte für das ausgesuchte Protokoll nicht relevant sind, da es bestimmte Funktionalitäten nicht gibt oder Funkprotokolle eine andere Vorgehensweise verwenden.

Nach der Auswahl des Funkprotokolls Bluetooth Low Energy sind zwei der aufgelisteten Vorgehensweisen für das Protokoll nicht relevant, sodass diese bei der Analyse nicht berücksichtigt werden. In BLE gibt es keine Befehle, die eine bestimmte Nähe zum Gerät voraussetzen, um ausgeführt zu werden. Somit entfällt dieser Punkt in der Sicherheitsanalyse, der beispielsweise in ZigBee relevant gewesen wäre, da es dort bestimmte Funktionalitäten gibt, bei denen sich der Sender im direkten Umkreis des Geräts befinden muss. Bei den folgenden Analysen der BLE-Geräte wird dieser daher nicht untersucht.

Die Untersuchung der Verfügbarkeit der Geräte und der drahtlosen Verbindung sind bei Bluetooth Low Energy ebenfalls nicht besonders relevant, da nach dem erfolgreichen Verbinden zweier Geräte durchgehend sogenannte „Connection Events“ ausgetauscht werden. Der Master in der Verbindung sendet Pakete an den Slave und dieser antwortet [Heydon 2013, S.96f.]. In der Verbindung gibt es also dauerhafte Kommunikation. Wenn die Verbindung jedoch durch Störgeräusche oder zu große Entfernung der Kommunikationspartner zueinander gestört ist oder verloren geht, findet dieser Austausch nicht mehr statt und die Verbindung wird nach Ablauf einer festgelegten kurzen Zeit beendet. Somit werden auch keine Befehle mehr verschickt, die ggf. nicht ankommen könnten. Dieser Punkt ist also bei der Überprüfung bei BLE ebenfalls nicht relevant.

Störgeräusche könnten durch „btlejack“ erzeugt werden, um die Verbindung zwischen zwei Geräten zu beenden, um z.B. eine eigene Verbindung aufzubauen.

6 Anwendung der Sicherheitsanalyse

Mit der zuvor ausgewählten Hardware und Software kann nun die vorgeschlagene Sicherheitsanalyse an Geräten der Heimautomatisierung durchgeführt werden. Dafür wurden möglichst verschiedene Geräte ausgewählt. Zum einen sind dies gewöhnliche Produkte aus dem Bereich der Heimautomatisierung und zum anderen Produkte, die in sicherheitskritischen Bereichen eingesetzt werden können, um bestimmte Zugänge vor Zugriff durch unberechtigte Personen zu schützen. Bei den gewöhnlichen Produkten handelt es sich um Lampen und Heizkörperthermostate, bei den Sicherheitsprodukten um Schlösser.

6.1 Magic Blue Bulb

Bei der Magic Blue Bulb handelt es sich um eine smarte Lampe, die per Bluetooth Low Energy gesteuert werden kann. Vom Aussehen her ist die Lampe nicht von einer üblichen, nicht smarten Lampe zu unterscheiden. Sie ist jedoch in der Lage BLE-Befehle zu verarbeiten und die verbaute LED ermöglicht es in den Farben eines komplettes Farbspektrum sowie beliebigen Helligkeiten zu leuchten. Die Farbe und Helligkeit kann dabei durch den Benutzer nach seinen Wünschen eingestellt werden.

Die Lampe ist in der Abbildung 16 dargestellt, ein Screenshot der dazugehörigen App in Abbildung 17.

Abbildung 16: Lampe Magic Blue Bulb

Abbildung 17: App der Magic Blue Bulb

Der Benutzer bekommt in der App „LED Magic Blue“ (Android, Version 1.2.6) eine Übersicht über alle Magic Blue Bulbs in der näheren Umgebung und kann diese in seiner App hinzufügen sowie einen Namen vergeben. Danach stehen weitere Funktionen zur Verfügung, wie z.B. das Ein- und Ausschalten der Lampe per App, das Ändern der Farbe (siehe Screenshot in Abbildung 17), verschiedene voreingestellte Farben oder Farbänderungen nach bestimmter Musik, die auf dem mobilen Endgerät abgespielt wird, sowie die Möglichkeit einer automatischen Zeitschaltung. Ebenfalls ist es mit der App möglich mehrere Lampen zu gruppieren und diese gleichzeitig zu steuern. Diese Funktion ist allerdings auf vier Geräte pro Gruppe beschränkt. Die Lampe ist sofort nach der Installation der App mit dieser steuerbar, d.h. es wird lediglich in der App eine Suche gestartet, die alle Lampen dieser Art in Empfangsreichweite auflistet und die gewünschte Lampe kann dann vom Benutzer ausgewählt und

gesteuert werden.

Nachdem die Lampe physikalisch von Strom getrennt wird, geht diese aus und beim nächsten Einschalten des Stroms leuchtet sie weiter in der zuletzt eingestellten Konfiguration. Es kann sich maximal ein Benutzer gleichzeitig mit der Lampe verbinden und diese steuern.

6.1.1 Sicherheitsanalyse

Die Magic Blue Bulb soll mit der zuvor erarbeiteten Vorgehensweise auf ihre Sicherheit analysiert werden. Zunächst einmal werden dabei allgemeine Informationen über das Gerät erhoben, die eine genauere Analyse ermöglichen.

Bei der smarten Lampe Magic Blue Bulb wird zur Kommunikation zwischen App und Lampe Bluetooth Low Energy eingesetzt. Nach Anschließen der Lampe an eine Stromversorgung sendet diese automatisch in einem Intervall von 40-50 Millisekunden Advertising-Pakete, die Informationen über das Gerät enthalten. Einen Sleep-Modus, indem keine Daten gesendet und empfangen werden können gibt es nicht, da die Lampe durchgehend Advertising-Pakete aussendet und jederzeit mit einem Gerät verbunden werden kann. Es ist also nicht notwendig auf bestimmte Momente zu warten, um eine Verbindungsanfrage zu stellen. Mit Hilfe des Tools „BLEAH“ werden erste Eigenschaften über das Gerät übersichtlich zusammengefasst. Für die Analyse wichtige Informationen daraus sind beispielsweise die MAC-Adresse der Lampe oder der Adresstyp, der beim Verbinden mit der Lampe benötigt wird.

Der Benutzer der App benötigt keinen PIN oder ein Passwort um sich mit der Lampe zu verbinden, sondern ist in der Lage diese in der App, nachdem die Lampe eingeschaltet ist, auszuwählen und zu steuern. Somit werden auch keine PIN, Passwörter oder ähnliche sensible Daten benötigt und übermittelt.

Bei den Befehlen, die die App zur Lampe sendet, handelt es sich um Schreibbefehle um die Lampe zu steuern, besonders zu Beginn der Kommunikation werden einige Lesebefehle abgesendet. Da in der App die aktuellen Einstellungen der Lampe grafisch dargestellt werden (z.B. ein Marker auf dem Farbspektrum), müssen diese Einstellungen zunächst von der Lampe gelesen werden.

Mit Hilfe des Tools „BLEAH“ können weitere Informationen über die Services und Characteristics sowie die les- und schreibbaren Handle-IDs gewonnen werden. Demnach existieren verschiedene lesbare Characteristics wie beispielsweise der Gerätename, aber auch schreibbare. Die zuvor aufgenommenen Befehle zum Ändern der Farbe und Helligkeit der Lampe erfolgten auf die Characteristic mit dem Handle *0x000B*. Dieser ist in der Auflistung als einziges als schreibbar dargestellt. Über diesen Handle können also Schreibbefehle an die Lampe gesendet werden.

Der nächste Punkt des Vorgehens analysiert die Verschlüsselung und den Austausch sowie die Verteilung von Schlüsseln. Bluetooth Low Energy bietet an sich die Möglichkeit Verschlüsselung einzusetzen, in der Magic Blue Bulb wird darauf jedoch verzichtet. Dies ist durch das Mitschneiden und Analysieren des Verbindungsaufbaus mit der Lampe zu erkennen. Beim Pairing-Prozess in Bluetooth Low Energy kann Verschlüsselung initiiert werden, indem ein „Encryption Request“ („LL_ENC_REQ“) gesendet wird. Dieser Vorgang wird in [Heydon 2013, S. 112f.] genauer beschrieben. Nach der Anfrage eines Geräts der Verbindung nach Verschlüsselung, antwortet das andere Gerät mit einem „Encryption Response“ („LL_ENC_RSP“) und im Folgenden werden dann für die Verschlüsselung erforderliche Daten ausgetauscht und die Schlüssel berechnet.

Der Verbindungsprozess zwischen App und Magic Blue Bulb geschieht jedoch ohne eine solche Anfrage nach Verschlüsselung und somit findet die Kommunikation unverschlüsselt statt. Es gibt in diesem Fall also keine Verschlüsselung und es sind daher auch keine Schlüssel notwendig, die fest einprogrammiert sein könnten oder ausgetauscht werden. Entsprechend

ist die Kommunikation nicht vor Mitlesen geschützt. Somit sind Rückschlüsse auf die Befehle und Inhalte der Nachrichten relativ einfach möglich.

Im Punkt Integrität der Vorgehensweise geht es darum, Befehle zu manipulieren und ggf. eigene Befehle abzusenden, ohne dass erkannt wird, dass es sich dabei nicht um einen legitimen Absender handelt. Zunächst einmal muss dafür verstanden werden, wie die gesendeten Befehle aufgebaut sind und welche Funktion diese haben. Dafür wird die Kommunikation zwischen App und Lampe beim Verändern von Farbton und Helligkeit und den weiteren möglichen Funktionalitäten in der App aufgenommen und analysiert.

Beim Mitschneiden der Kommunikation fällt auf, dass es für das Ein- bzw. Ausschalten der Lampe jeweils einen statischen Schreibbefehl auf das Handle *0x000B* gibt. Dieser beginnt in beiden Fällen mit „CC“ und endet mit „33“, im Fall des Einschaltens befindet sich dazwischen das Byte „23“ und im Fall des Ausschaltens das Byte „24“. Außerdem fällt auf, dass beim Ändern der Farbe und/oder Helligkeit immer ähnlich lange und gleich aussehende Schreibbefehle ebenfalls auf das Handle *0x000B* abgesendet werden. Sie beginnen alle mit einem sich nicht ändernden Teil („56“) und enden mit einem ebenfalls statischen Teil („00 F0 AA“). Im Bereich zwischen diesen festen Teilen befinden sich immer genau drei Byte, die nach Einstellen bestimmter Farben in der App darauf schließen lassen, dass es sich dabei um einen RGB-Farbwert handelt. Einen gesonderten Wert oder Schreibbefehl für das Verändern der Helligkeit gibt es nicht, d.h. Helligkeit und Farbton werden über den RGB-Wert festgelegt. Ähnlich sehen die Befehle aus, die gesendet werden, wenn das Licht nach den am Handy aufgenommenen Lautstärken eingestellt werden soll. Der Befehl ist der gleiche wie beim Ändern des Farbtons und der Helligkeit mit dem Unterschied, dass dieser nicht mit „56“ beginnt, sondern mit „78“. Bei den Sonderfunktionen, die in der App möglich sind (z.B. Stroboskop oder Farbverläufe), gibt es ebenfalls einen relativ einfachen Befehlsaufbau. Ein fester Wert „BB“ wird gefolgt von einem Wert im Hexadezimalbereich von 26 bis einschließlich 38, gefolgt von der Geschwindigkeit der Funktion und einem festen Wert „44“. Alle genannten Befehle sind im Detail im Anhang in II.1 aufgelistet.

Da die Kommunikation unverschlüsselt erfolgt, können die Befehle des Benutzers mitgelesen werden. Es können außerdem aufgenommene Befehle Befehle zufällig und systematisch verändert werden sowie mit dem Wissen über den Aufbau der Nachrichten eigene Befehle erstellt werden.

Zum Erstellen und Absenden der oben genannten Befehle wird ein Python-Script verwendet, das das Framework „bluepy“ verwendet. Da es keine Mechanismen gibt, um die Befehle zu schützen, können diese einfach erstellt und gesendet werden. Alle oben genannten Befehle wurden von der Lampe verarbeitet und ohne Probleme durchgeführt. Somit ist die Lampe vollständig durch jeden in Bluetooth-Reichweite steuerbar, auch ohne die zugehörige App zu verwenden. Allerdings können alle Einstellungen durch den legitimen Benutzer wieder überschrieben werden. Es ist nicht notwendig aufgenommene Befehle zufällig zu verändern (Fuzzing) oder beliebige Befehle zu senden, da bereits alle Funktionen an dieser Stelle ohne Probleme nachstellbar sind.

Eine weitere Fragestellung ist die, ob es eine Integritätsprüfung gibt und worauf diese abzielt. Alle Befehle haben am Ende zwar einen weiteren Befehlsteil, der zu dem Befehl an sich (z.B. Farbwert einstellen) unterschiedlich ist. Jedoch ist dieser nur für die jeweilige Art von Befehl unterschiedlich, nicht für jeden einzelnen Befehl. Bei diesem Teil des Befehls handelt es sich somit um einen festen Wert pro Befehlsart und nicht um eine Prüfsumme, denn diese wäre mit jedem Befehl (z.B. anderer RGB-Wert beim Befehl zum Ändern der Farbe) unterschiedlich.

Außerdem wurden verschiedene gültige Befehle verändert, d.h. feste Werte weggelassen, verändert sowie komplett zufällige Werte generiert, und abgesendet. Das Absenden von rein zufälligen Werten als Nachrichteninhalte wird jedoch ignoriert, beim Einspielen fehlerhafter

Nachrichten reagiert die Lampe also darauf nicht. Danach ist die Lampe wieder wie zuvor über die bekannten Befehle und die App steuerbar.

Da die Magic Blue Bulb nicht mit einer Batterie oder einem Akku betrieben wird, sondern über eine ständige Stromversorgung per Kabel verfügt, kann die Stromversorgung höchstens unterbrochen werden, wenn der Benutzer einen Schalter umlegt oder der Strom ausfällt. Für Angreifer gibt es also keine Möglichkeit diese mittels Absenden vieler Befehle oder Ähnlichem abzuschalten. Der Benutzer erfährt vom Ausfall der Lampe, indem bei bestehender Verbindung die Verbindung nach Ablauf einer festgelegten Zeit beendet wird. In der App äußert sich dies so, dass der Benutzer wieder auf die Übersichtsseite geleitet wird. Nach dem Neustart des Geräts bzw. nach erneutem Einschalten der Stromversorgung wird die zuletzt eingestellte Kombination aus Farbe und Helligkeit eingestellt und die Lampe angeschaltet, unabhängig davon, ob sie zuvor per App an- oder ausgeschaltet war. Weitere Konsequenzen gibt es nicht, nach einer neuen Verbindungsanfrage ist die Lampe wie zuvor steuerbar.

Als nächstes wird der Punkt der Authentizität untersucht. Der Benutzer wird generell nicht nach Benutzerdaten wie Passwörtern gefragt und es werden zur Steuerung der Lampe auch keine solcher Daten benötigt. Das bedeutet, dass die Lampe keine Abhängigkeiten zu Benutzern in den Befehlen beinhaltet und die Befehle für alle Absender gleich sind. Authentizität wird also nicht verlangt, jeder Befehl jedes Absenders wird angenommen und ausgeführt.

Ein Zurücksetzen des Geräts auf den Werkszustand ist bei der Magic Blue Bulb nicht relevant, da keine Eigenschaften zurücksetzbar sind und keine bestimmten Werte gespeichert werden. Lediglich der zuletzt eingestellte Farbwert bleibt beim Neustart erhalten, ändert sich jedoch, sobald ein neuer Farbwert gesetzt wird.

Ein weiterer Teil der Analyse beinhaltet die Frage wie sicher Update-Mechanismen sind. In der App der Magic Blue Bulb konnte keine Möglichkeit gefunden werden die Firmware der Lampe zu updaten. Somit gibt es auch keine Möglichkeit die nicht vorhandenen Sicherheitsmechanismen nachträglich zu korrigieren. Da Updates nicht vorgesehen sind, erübrigen sich auch die Fragen nach Authentizität und Verschlüsselung von Updates sowie nach dem Einspielen von Downgrades. Die aktuelle Firmware-Version der Lampe kann in der App ebenfalls nicht eingesehen werden.

Das letzte zu analysierende Schutzziel ist die Nicht-Vermehrbarkeit. Dabei ist zunächst die Fragestellung, ob es möglich ist Nachrichten aufzunehmen und zu einem späteren Zeitpunkt wieder einzuspielen. Bei der Magic Blue Bulb ist dies möglich. Sämtliche Befehle werden unverschlüsselt versendet und entsprechend resultiert aus jeder ausgeführten Aktion in der App der jeweils gleiche Befehl. Schutzmechanismen vor Wiedereinspielen wie beispielsweise der Einsatz eines Counters, Zeitstempeln oder einmalig gültige Werte gibt es nicht. Somit ist es ohne Probleme möglich zuvor aufgenommene Nachrichten wieder einzuspielen und die Lampe reagiert entsprechend auf diese Befehle, als wären sie von einem legitimen Absender gesendet worden.

Zur Identifizierung von Geräten werden die MAC-Adressen der Geräte verwendet. Da die smarte Lampe jedoch sämtliche Verbindungen zulässt und es auch ohne weitere Schutzmechanismen für jedes Gerät ermöglicht Befehle zu senden, ist es nicht notwendig Geräte zu fälschen, um an Daten zu kommen oder Befehle zu senden. Sämtliche Daten sind ohne Probleme ohnehin schon abrufbar und alle Befehle sind absendbar.

6.1.2 Ergebnis

Das Ergebnis der durchgeführten Sicherheitsanalyse wird in der Tabelle 1 aufgeführt und im Folgenden zusammengefasst.

Tabelle 1: Zusammenfassung Analyse Magic Blue Bulb

Funkprotokoll	Bluetooth Low Energy (BLE)
Verbindungsaufbau	Immer möglich bei Stromversorgung
Stromversorgung	Dauerhaft (Kabel)
Verschlüsselung	Nein
Schlüsselaustausch	Nein
Nachrichtenlänge/-aussehen	Rückschlüsse auf Inhalt möglich
Befehle wiedereinspielbar	Ja
Befehle veränderbar	Ja, vollständig nachstellbar
Fehlerhafte Nachrichten	Werden ignoriert
Integritätsprüfungen	CRC
Verhalten nach Neustart	Lampe schaltet sich ein mit letzter Einstellung
Authentifikation	Nein
Firmware Up-/Downgrades	Keine Updates möglich

Wie bereits aus der Tabelle hervorgeht, ist die Magic Blue Bulb, sobald sie mit dem Strom verbunden ist, bereit Verbindungen entgegenzunehmen, was durch Aussenden von Advertisement-Nachrichten nach außen vermittelt wird. Sämtliche Kommunikation erfolgt unverschlüsselt und ist somit für jeden Angreifer in der Nähe mitlesbar. Dadurch sind Rückschlüsse auf alle gesendeten Befehle möglich und es ist ebenfalls möglich, dass Angreifer alle aufgenommenen Befehle erneut einspielen können. Vielmehr ist es möglich, dass diese Befehle selbst erstellen können und diese von der Lampe verarbeitet werden. Dabei muss es sich allerdings um gültige Befehle handeln, fehlerhafte Befehle werden ignoriert. Neben der nicht vorhandenen Verschlüsselung existiert auch keine Integritätsprüfung, lediglich die von BLE vorgesehene CRC-Prüfsumme am Ende jeder Nachricht ist vorhanden, diese schützt jedoch nur vor Übertragungsfehlern und nicht vor Veränderung von Nachrichten. Authentifikation von Benutzern ist ebenfalls nicht gegeben, jeder Benutzer und jeder Angreifer in der Nähe kann die Lampe steuern, wenn er Kenntnis über die Befehle besitzt. Eine Funktionalität um Firmware-Updates einzuspielen ist nicht vorgesehen. Ein Neustart der Lampe lässt diese wieder in die letzte Konfiguration übergehen.

Zusammenfassend existieren bei der Magic Blue Bulb keine Maßnahmen, um einen Angreifer davon abzuhalten die Lampe vollständig steuern zu können. Bluetooth Low Energy an sich besitzt einige Maßnahmen, die einen Zugriff und das Steuern durch Dritte verhindern könnten, diese werden jedoch nicht genutzt.

6.2 Tapplock One

Das Tapplock One ist ein smartes Vorhängeschloss, das üblicherweise verwendet werden kann, um den Zugriff unberechtigter Personen zu bestimmten Gegenständen oder Orten zu verhindern.

Optisch wirkt das Schloss auf den ersten Blick wie ein handelsübliches Vorhängeschloss, besitzt jedoch ein etwas dickeres Gehäuse. Es besitzt zudem keine Öffnung, um das Schloss

mit einem Schlüssel öffnen zu können, sondern auf der Vorderseite des Gehäuses einen Fingerabdrucksensor sowie eine kleine Kontrollleuchte, die in verschiedenen Farben leuchten kann (unterschiedliche Bedeutungen je nach Farbe).

Die Abbildung 18 zeigt das Schloss in geschlossenem Zustand. In der Mitte befindet sich der Fingerabdrucksensor, darüber die Kontrollleuchte und an der Unterseite des Schlosses ein kleiner Knopf. Da das Schloss physikalisch vor gewaltsamem Öffnen geschützt sein soll, gibt es an der Unterseite des Schlosses außerdem einen Anschluss für ein Ladekabel, womit der Akku aufgeladen werden kann, somit muss das Gehäuse nicht geöffnet werden, z.B. um diesen auszutauschen.

Abbildung 18: Vorhängeschloss Tapplock One

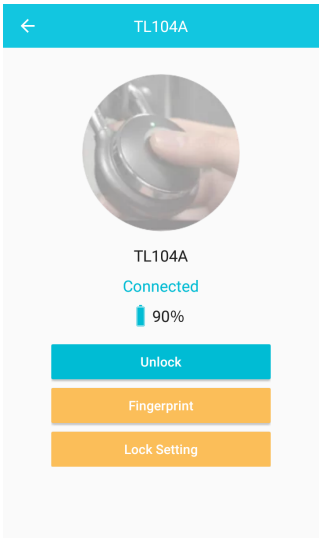

Abbildung 19: App des Tapplock One

Der Besitzer des Schlosses hat drei Möglichkeiten das Schloss zu Öffnen: App, Fingerabdrucksensor und Morse-Code. In der zugehörigen App existiert die Möglichkeit sich mit dem Schloss per Bluetooth Low Energy zu verbinden und verschiedene Einstellungen vorzunehmen, wie beispielsweise das Einstellen eines Fingerabdrucks, mit dem das Schloss geöffnet werden soll oder eines Morse-Codes. Der Benutzer kann das Schloss also mit der App öffnen, mit dem Fingerabdrucksensor oder per Morse-Code, der mit einer kleinen Taste unterhalb des Schlosses eingegeben wird.

Das Schloss wird mit einem eingebauten Akku betrieben, der auch mit einem mitgelieferten Ladekabel wieder aufgeladen werden kann. Der Akku ist laut Hersteller dafür ausgelegt im Normalfall für eine Stromversorgung des Schlosses für bis zu einem Jahr zu sorgen.

In der dazugehören App „Tapplock“ (Android, Version 2.3.0) kann der Benutzer nach Erstellen eines Benutzeraccounts bei Tapplock und Anmelden mit diesem alle Tapplock-Schlösser in seiner Reichweite auflisten lassen und auswählen. Bei Auswahl eines Schlosses wird die Verbindung hergestellt und der Benutzer bekommt den aktuellen Ladezustand des Akkus angezeigt und kann das Schloss öffnen, Fingerabdrücke eingeben oder weitere Einstellungen vornehmen, wie z.B. das Update der Firmware. Um das Schloss zu öffnen ist lediglich ein Klick auf den entsprechenden Button notwendig, das Schloss öffnet sich sofort und zeigt dies auch mit Leuchten der LED in grüner Farbe an.

6.2.1 Sicherheitsanalyse

Das Tapplock One ist das Schloss, das auch im Decompile Angriff in 3.9 beschrieben wird. Dementsprechend ist bereits bekannt, dass dieses Schloss sicherheitskritische Lücken bein-

haltet, die es ermöglichen das Schloss zu öffnen. Die Autoren beschreiben in [Tierney 2018a], dass es möglich ist allein mit Wissen über die MAC-Adresse des Schlosses dieses zu öffnen.

Eine Recherche auf der Homepage des Herstellers ergab, dass die Sicherheitslücke dem Hersteller mitgeteilt wurde und dieser ein Update veröffentlicht hat, das diese schließen soll [Tapplock 2018]. Benutzer sollen sowohl ihre App, als auch die Firmware des Schlosses updaten. Das vorhandene Tapplock Schloss befindet sich noch im Werkszustand und wurde bisher nicht mit der App verbunden. Es besitzt also nicht die aktualisierte Firmware und müsste somit noch verwundbar gegen die in [Tierney 2018a] beschriebene Vorgehensweise sein. Aus diesem Grund wird zunächst darauf verzichtet das Schloss mit der App zu verbinden, wodurch der Angriff unter Umständen unmöglich wird, wenn automatisch Updates durchgeführt werden. Zunächst soll der beschriebene Angriff durchgeführt werden, um die - zumindest frühere - Vorgehensweise des Schlosses nachvollziehen zu können. Im Anschluss wird dann die App installiert und das Schloss mit dieser verbunden, um die Sicherheitsanalyse durchzuführen.

Der beschriebene Angriff besteht darin, dem Schloss zwei Befehle zu senden, um es zu öffnen. Zum einen wird zunächst eine Verbindung mit Bluetooth Low Energy zum Schloss aufgebaut und ein Befehl gesendet, der es dem Benutzer erlaubt weitere Befehle auszuführen. Der Befehl wird von den Herstellern als Pairing-Befehl bezeichnet, ist jedoch unabhängig vom Pairing-Prozess, der in BLE vorgesehen ist. Dieser erste Befehl setzt sich aus mehreren Teilen zusammen: einem festen ersten Teil, der bei jedem Schloss gleich ist und als „Regular Pair“ bezeichnet wird. Er wird gefolgt von den gerätespezifischen Variablen „Key1“ und „SerialNo“, die Teile eines Hashwertes der MAC-Adresse des Schlosses sind. Der zweite Befehl ist der Befehl zum Öffnen des Schlosses. Er besteht lediglich aus einem statischen Wert. Beide Befehle werden per Write-Befehl an das Schloss gesendet und laut Autoren öffnet sich dieses dann.

Zum Nachstellen dieses Angriffs wird zunächst die MAC-Adresse des Schlosses benötigt, um die beiden gerätespezifischen Variablen zu berechnen, die für den ersten Befehl benötigt werden. Somit ist der erste Punkt in der Sicherheitsanalyse ohnehin erforderlich, um Informationen über das Gerät zu erlangen.

Das Schloss sendet generell keine Advertising-Pakete dauerhaft aus und ist somit zunächst auch nicht zu finden. Erst nach zweimaligem Betätigen des kleinen Knopfes an der Unterseite des Schlosses, beginnt dieses Advertising-Pakete im Intervall von etwa 40-50 Millisekunden auszusenden und ist somit für andere Bluetooth-Geräte sichtbar. Bei einmaligem Drücken des Knopfes leuchtet die LED weiß, das Gerät wäre nun bereit einen Fingerabdruck entgegenzunehmen, bei zweimaligem Drücken blinkt die Lampe blau und das Gerät ist bereit sich mit anderen Geräten zu verbinden und Bluetooth Nachrichten zu empfangen. Dieser Modus hält für 20 Sekunden an, danach geht das Schloss wieder in den Sleep-Modus. Dadurch wird Energie in der Batterie eingespart, da das Gerät weder Nachrichten senden, noch empfangen können muss. Gleichzeitig beinhaltet diese Funktionalität jedoch auch den Nebeneffekt, dass für die Steuerung des Schlosses auch ein physikalischer Zugang zu diesem notwendig ist und dieses nicht zu jedem beliebigen Zeitpunkt aus der Ferne gesteuert werden kann.

In der Übersicht in „BLEAH“ ist neben Eigenschaften wie der MAC-Adresse auch erkennbar, dass es verschiedene Characteristics gibt, auf denen Lese- bzw. Schreibzugriffe möglich sind. Für das Aussenden von Befehlen sind die Schreibzugriffe von Bedeutung, beispielsweise sind diese auf den Gerätenamen möglich oder auch auf das Handle *0x000E*. Die gewonnenen Informationen sind wichtig, um den Angriff aus [Tierney 2018a] nachzustellen.

Mit der MAC-Adresse des Schlosses können die Variablen „Key1“ und „SerialNo“ berechnet werden, indem die MAC-Adresse invertiert wird, d.h. die Byte-Blöcke werden umgekehrt aufgelistet und mit einem Doppelpunkt miteinander verbunden. Alle Buchstaben werden in Großbuchstaben umgewandelt und ein MD5-Hashwert wird gebildet. „Key1“ sind dabei

ersten acht Zeichen des Hashwertes und „SerialNo“ die letzten acht.

Der erste feste Befehlsteil wurde aus den Screenshots in [Tierney 2018a] entnommen und der zweite Teil wurde selbst berechnet, da er sich für das vorhandene Schloss unterscheidet. Allerdings gibt es am Ende des fertigen Befehls in den Screenshots noch vier weitere Zeichen, die nicht näher erklärt sind. Da diese in den Screenshots in der Arbeit und einem Video, das zu dem Angriff auf das Schloss existiert, jedoch unterschiedlich sind, ist es zunächst erforderlich, herauszufinden welche Bedeutung diese Zeichen haben.

Aus diesem Grund wurde die Android-App für das Schloss dekompiliert und näher untersucht. Dies war ohne weitere Probleme mit entsprechenden Online-Tools (Jadx-Decompiler) möglich und die dekompilierten Dateien waren gut und einfach zu lesen sowie der Code gut zu verstehen. Dabei fiel auf, dass alle Befehle zusätzlich eine zwei Byte (4 Zeichen hexadezimal) große Prüfsumme bekommen, die an die Befehle angehängt wird. Dabei handelt es sich nicht um die Prüfsumme, die ohnehin in Bluetooth Low Energy nach den Befehlen vorgesehen ist (CRC-Prüfsumme), sondern eine eigene weitere Prüfsumme des Herstellers innerhalb des abgesendeten Befehls, also als Teil der Nutzdaten. Somit wurde die Berechnung der Prüfsumme ebenfalls in einem Python-Script (mit Hilfe der Berechnungen im Quellcode der App) für den zuvor erstellten Befehl durchgeführt, sodass der fertige Befehl zum Verbinden mit dem Schloss erstellt werden konnte. Da der Befehl zum Öffnen des Schlosses keine variablen Teile enthält, konnte dieser inklusive der Prüfsumme aus den Screenshots übernommen werden.

Zum Senden der beiden Befehle wurde ein weiteres Python-Script geschrieben, das Befehle aus dem Bluetooth Low Energy Interface „bluepy“ verwendet. Es wird zunächst eine Verbindung mit dem Schloss aufgebaut und dann beide Befehle hintereinander als Schreibbefehle auf das Handle *0x000E* abgesendet, das einen Schreibzugriff erlaubt. Zur Kontrolle wird die Kommunikation mit dem Texas Instruments (TI) Packet Sniffer mitgeschnitten. Das Schloss öffnet sich entgegen der Erwartungen jedoch nicht. Die Verbindung kommt zwar zu Stande - wie auch im Mitschnitt erkennbar ist -, jedoch leuchtet die LED nicht durchgehend blau, sondern bleibt im Verbindungsmodus (blau blinkend), sodass daraus gefolgert werden kann, dass bereits der erste Befehl zum Verbinden („Regular Pair“) nicht angenommen wird.

Der dekompilierte Code der App wurde daher näher analysiert, insbesondere im Hinblick auf den Verbindungsaufbau bei einem Schloss im Werkszustand. In diesem Fall wird nicht die Methode „Regular Pair“ aufgerufen, die in der Arbeit von [Tierney 2018a] beschrieben wird, sondern eine andere Methode „First Pair“, in der lediglich ein fester vorgegebener Befehl zusammen mit dem Wert *AE9566C7* an das Schloss gesendet wird. Beim Senden dieses Befehls wird die Verbindungsanfrage an das Schloss angenommen und auch die LED leuchtet durchgehend blau, sodass die Verbindung erfolgreich ist. Das Senden des Öffnen-Befehls im Anschluss bleibt jedoch immer noch wirkungslos.

In der App folgt nach dem erstmaligen Verbinden mit dem Schloss mit dem entsprechenden Befehl noch ein weiterer Befehl „Boot“. Erst danach ist das Schloss im Normalzustand wie es zum Sichern von Gegenständen verwendet werden kann und dieser bleibt bestehen. Nach dem Senden des Befehls „First Pair“ und anschließend des Befehls „Boot“ ist es möglich den in [Tierney 2018a] beschriebenen Angriff durchzuführen. Es wird also der zusammengesetzte erste Befehl aus den Variablen der MAC-Adresse gefolgt vom Befehl zum Öffnen des Schlosses abgesendet und das Schloss öffnet sich.

Somit ist es in der Tat möglich - zumindest bei Verwendung des Schlosses im Auslieferungszustand ohne Updates -, das Schloss lediglich mit Wissen der MAC-Adresse zu öffnen, was eine enorme Sicherheitsproblematik darstellt, da ein Schloss dafür ausgelegt ist Gegenstände oder Räume vor dem Zugriff Unbefugter zu schützen.

Die Abbildung 20 veranschaulicht die gesamte Vorgehensweise in einem Diagramm. Nur bei Schlössern, die sich nicht mehr im Werkszustand befinden und bereits mit der App verbunden sind, reicht es aus die Vorgehensweise ab „REGULAR_PAIR“ durchzuführen,

ansonsten sind alle der aufgezeigten Schritte notwendig.

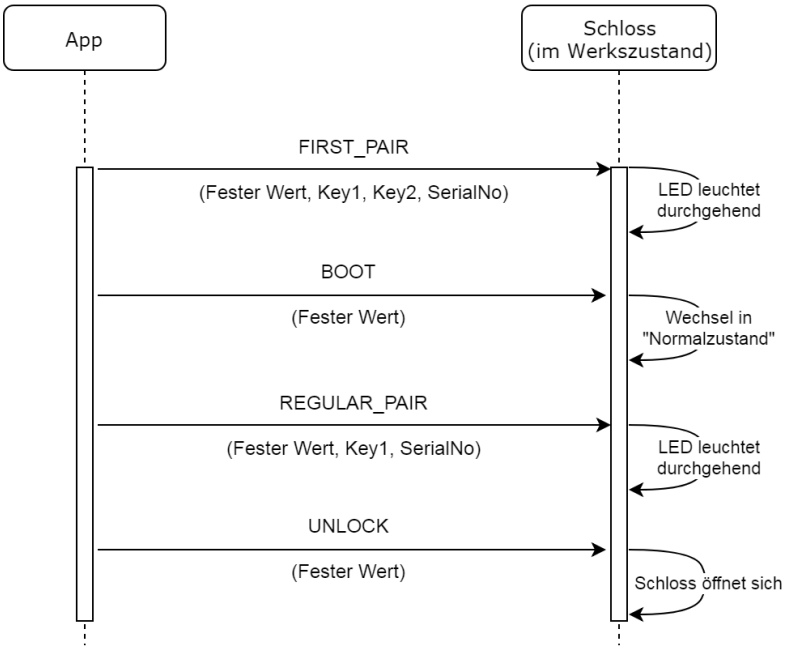

Abbildung 20: Vorgehensweise zum Öffnen des Schlosses im „Normalzustand“

Im Quellcode der App gibt es neben dem Öffnen noch weitere Befehle, die möglich sind, nachdem das Verbinden mit dem Schloss erfolgreich war. Einer dieser Befehle ist das Zurücksetzen auf den Werkszustand („Factory Reset“). Die Autoren in [Tierney 2018a] beschreiben zwar es gäbe keinen Befehl zum Zurücksetzen des Schlosses auf den Werkszustand, der es ermöglicht erteilte Zugriffe rückgängig zu machen. Das ist soweit auch korrekt, da der Befehl zum Verbinden und Öffnen ja nicht abhängig ist von benutzerspezifischen Daten, sondern lediglich von der MAC-Adresse des Schlosses. Es ist aber sehr wohl möglich, das Schloss auf den Werkszustand zurückzusetzen. Dieser bewirkt, dass vor dem Öffnen wieder ein „First Pair“ notwendig ist gefolgt von einem „Boot“-Befehl, damit das Schloss den Befehl „Regular Pair“ entgegennimmt. Das Schloss wird also mit dem entsprechenden Befehl wieder in den Werkszustand zurückgesetzt, sodass es mit der App verbunden werden kann, um die weitere Sicherheitsanalyse durchzuführen. Alle zuvor genannten Befehle für das vorhandene Schloss sind im Detail im Anhang in II.2 aufgelistet.

Nachdem das Schloss bereits geöffnet werden konnte, werden nun der Vollständigkeit halber die weiteren Punkte der Analyse mit Hilfe der App behandelt. Der erste dieser Punkte ist die Vertraulichkeit, der sowohl die Initiierung der Verschlüsselung, als auch die Verschlüsselung an sich beinhaltet. Beim Tapplock One wird nach dem Verbinden mit dem Schloss jedoch kein Schlüsselaustausch initiiert. Dies ist daran zu erkennen, dass kein „Encryption Request“ gesendet wird. Somit erfolgt die Kommunikation also nicht mit dem im BLE-Standard vorgesehenen Pairing und der anschließenden Verschlüsselung. Da die Befehle aus dem dekompilierten Code bereits bekannt sind, ist zu erkennen, dass auch kein eigenes Verfahren zur Verschlüsselung verwendet wird. Die Befehle werden im Klartext an das Schloss gesendet. Es gibt also auch keine weiteren fest einprogrammierten Schlüssel oder eine Übertragung von Schlüsseln per Bluetooth Low Energy.

Beim Auswählen des Schlosses in der App wird zunächst eine Verbindung mit dem Schloss hergestellt und verschiedene Handles abgefragt sowie Schreibbefehle abgesendet, um beispielsweise den Ladestatus des Akkus zu erfahren, der dann in der App angezeigt wird.

Außerdem wird der Befehl „Regular Pair“ gesendet, sodass die LED am Schloss durchgehend blau leuchtet und der Benutzer weitere Befehle senden kann. Dies kann beispielsweise der Befehl zum Öffnen des Schlosses sein.

Da alle Befehle unverschlüsselt versendet werden, ist es möglich diese mitzulesen und da diese unterschiedliche Längen haben und mit unterschiedlichen festgelegten Zeichenfolgen beginnen, ist es möglich Rückschlüsse auf die Inhalte der Befehle zu ziehen.

Das nächste Schutzziel ist die Integrität. Die möglichen Befehle sind ähnlich zueinander aufgebaut. Nachdem die Verbindung mit dem Schloss aufgebaut ist, ist ein Befehl notwendig, damit das Schloss weitere Befehle entgegennimmt, was durch das durchgehende Leuchten der blauen LED am Schloss zu erkennen ist. Im Werkszustand ist das ein fester Befehl („First Pair“), im verschlossenen Zustand ist dies ein fester Befehl („Regular Pair“) jeweils gefolgt von Bestandteilen eines MD5-Hashwertes der MAC-Adresse. Danach sind weitere Befehle möglich, die in der Regel lediglich aus einem festen Wert bestehen. Durch das Dekompilieren der Android-App war es möglich eine Übersicht über alle möglichen Befehle zu erhalten. Um das Schloss zu öffnen, reicht es jedoch aus lediglich den Unlock-Befehl zu kennen. Bei einem Schloss im Werkszustand ist zusätzlich das Wissen über den Boot-Befehl notwendig. Alle Befehle werden als Write-Befehl an das Handle *0x000E* gesendet.

Beim Einspielen sämtlicher Nachrichten bevor ein Verbindungsbefehl („Regular Pair“ bzw. „First Pair“ im Werkszustand) durchgeführt wurde, werden diese verworfen. Gleiches geschieht bei fehlerhaften Nachrichten (bei inhaltlich fehlerhaften und bei falschen Prüfsummen) nach einem gültigen Verbindungsbefehl. Fehlerhafte Nachrichten werden also ignoriert.

Neben der im BLE-Protokoll vorgesehenen CRC-Prüfsumme, die lediglich der Entdeckung von Übertragungsfehlern dient, gibt es beim Tapplock One eine weitere eigene Prüfsumme mit einer Länge von zwei Byte am Ende aller Befehle. Der Algorithmus dieser Prüfsumme konnte im dekompilierten Code entdeckt werden und in einem Python-Script umgeschrieben werden, sodass diese automatisch für die selbst erstellten Befehle berechnet werden kann. Das Schloss überprüft beim Empfang der Nachrichten, ob diese zusätzliche Prüfsumme korrekt ist. Ein Weglassen oder verändern dieser Prüfsumme führt dazu, dass das Schloss den Befehl verwirft. Dadurch, dass keine Verschlüsselung verwendet wird oder andere Sicherheitsmaßnahmen getroffen sind, kann diese Prüfsumme manipuliert werden - in diesem Fall selbst berechnet und an beliebige Nachrichten angehängt werden.

Ein Neustart durch das Leeren der eingebauten Akkus (nach erneutem Aufladen per Ladekabel) sorgt für keine weiteren Konsequenzen durch Zurücksetzen flüchtiger Variablen, da es keine sicherheitsrelevanten Variablen wie z.B. Counter gibt. Die Konsequenzen beschränken sich also lediglich darauf, dass beim Entleeren der Akkus der legitime Benutzer nicht mehr in der Lage ist sein Schloss zu öffnen, bis er es erneut aufgeladen hat.

Im Punkt der Authentizität gibt es zwar Benutzerdaten, die der Benutzer eingibt, um sich in der App einzuloggen. Für die Kommunikation mit dem Schloss ist jedoch keine Authentifizierung notwendig. Wie zuvor beschrieben, wird jede Verbindung zugelassen und jeder Befehl wird zugelassen, sofern vorher ein gültiger Befehl „Regular Pair“ gesendet wurde. Dieser Befehl reicht aus, um den Benutzer als legitimen Benutzer zu authentifizieren. Natürlich ist das Absenden eines solchen Befehls mit Kenntnissen über den Inhalt auch für jeden beliebigen Angreifer möglich. Authentizität ist hier also nicht gegeben.

Tapplock bietet die Möglichkeit von Firmware-Updates an. Dafür gibt es in den Einstellungen zum Schloss in der App einen Button, der prüft, ob es eine neue Firmware-Version gibt und diese dann per Bluetooth Low Energy auf das Schloss überträgt. Laut der App steht zwar ein Firmware-Update zur Verfügung, das nach einer Bestätigung übertragen und installiert werden könnte, da aber das Schloss zu Schulungszwecken mit der entsprechenden unsicheren Steuerung noch im aktuellen Zustand erhalten bleiben soll, wird darauf verzichtet die Firmware zu aktualisieren, wodurch die Sicherheitslücke nach Herstellerangaben [Tapplock 2018]

behoben sein soll. Da das Schloss in seiner Firmware nicht verändert werden soll und auch keine veraltete Firmware-Version zur Verfügung steht, die übertragen werden könnte um zu überprüfen, ob nachträglich das Einspielen alter Firmware-Versionen (z.B. der aktuell installierten unsicheren Version 1.0) akzeptiert wird, kann nicht geprüft werden, ob es möglich ist veraltete Firmware-Versionen auf das Schloss einzuspielen.

Es gibt im Tapplock One keine Vorkehrungen um das erneute Einspielen von Nachrichten zu verhindern. Es werden keine Counter, Zeitstempel oder Nonces verwendet und jeder abgesendete Befehl ist immer gleich, d.h. statisch. Auch wenn es in der Android-App möglich ist, weiteren Benutzern Zugang zum Schloss zu geben, resultiert auch aus deren Aktion zum Öffnen der gleiche BLE-Befehl, da lediglich die MAC-Adresse des Schlosses im Verbindungsbefehl verwendet wird, die für alle Benutzer gleich ist und sich nur pro Schloss unterscheidet. Somit wäre es auch ohne Kenntnisse aus der Arbeit von [Tierney 2018a] darüber, dass lediglich ein MD5-Hashwert der MAC-Adresse ausreicht um das Schloss zu öffnen, möglich das Schloss zu öffnen. Es kann die Kommunikation aufgenommen werden und die immer gleichen Befehle erneut in der entsprechenden Reihenfolge eingespielt werden. Hier reicht es aus den Befehl zum Verbinden („Regular Pair“) und den Befehl zum Öffnen aufzunehmen, da dies ein Öffnen bewirkt, auch ohne zuvor Akkustatus oder andere Werte abgefragt zu haben.

Das Klonen oder Fälschen von Geräten ist nicht notwendig, da jede Verbindung jedes Geräts zugelassen wird und eine Identifizierung von legitimen Geräten nicht stattfindet. Jeder, der die MAC-Adresse des Schlosses kennt - und diese kann aus den Advertising-Paketen des Schlosses erhalten werden - ist in der Lage die entsprechenden Befehle auszusenden, unabhängig von seiner eigenen MAC-Adresse oder von anderen Merkmalen.

6.2.2 Ergebnis

Die Ergebnisse der Analyse werden in der Tabelle 2 zusammengefasst.

Tabelle 2: Zusammenfassung Analyse Tapplock One

Funkprotokoll	Bluetooth Low Energy (BLE)
Verbindungsaufbau	20 Sekunden nach Tastendruck möglich
Stromversorgung	Akku + Ladekabel
Verschlüsselung	Nein
Schlüsselaustausch	Nein
Nachrichtenlänge/-aussehen	Rückschlüsse auf Inhalt möglich
Befehle wiedereinspielbar	Ja
Befehle veränderbar	Ja, vollständig nachstellbar
Fehlerhafte Nachrichten	Werden ignoriert
Integritätsprüfungen	Eigene (manipulierbar), CRC
Verhalten nach Neustart	Letzter Zustand
Authentifikation	Nein
Firmware Up-/Downgrades	Updates manuell möglich, Downgrades nicht geprüft

Durch die fehlende Verschlüsselung und statischen Befehle ist es möglich gesendete Befehle aufzunehmen und wiedereinzuspielen, um das Schloss zu öffnen. Mit Kenntnis über den Aufbau der Befehle durch Dekompilieren der App oder Lesen des Artikels in [Tierney 2018a] ist es zudem möglich jedes beliebige Tapplock One zu öffnen, zumindest mit einer Firmware-Version im Auslieferungszustand. Die Prüfsumme am Ende der Befehle schützt nicht vor Manipulationen der Nachrichten, da diese ebenfalls manipuliert und selbst berechnet werden kann. Lediglich die MAC-Adresse des Schlosses muss bekannt sein um es zu öffnen und diese ist durch die Advertising-Pakete des Schlosses ermittelbar.

Das Mitschneiden und Wiedereinspielen von Nachrichten oder Senden von selbst zusammengestellten Nachrichten ist möglich. Durch Öffnen des Schlosses könnte ein Angreifer zudem Zugriff auf geschützte Gegenstände oder Räume bekommen, der durch das Schloss eben verhindert werden sollte. Somit ist das Schloss insgesamt als unsicher zu bewerten. Dabei handelt es sich um eine Analyse des Schlosses im Werkszustand, ein Schloss mit aktueller Firmware-Version wurde nicht untersucht.

6.3 Eqiva Türschlossantrieb

Der Eqiva Türschlossantrieb ist ein Gerät, das von innen auf dem Schloss einer Wohnungstür angebracht wird, um dieses mit einer App zu steuern. Dafür befindet sich im Gerät eine Öffnung, die über das Schloss, indem ein passender Schlüssel steckt, positioniert wird. In der Öffnung wird der Schlüssel fixiert und dieser wird dann über einen Motor gedreht, sodass dieser die Tür entweder abschließen, aufschließen oder öffnen kann. Betrieben wird das Gerät mit Batterien, die an der Seite des Türschlossantriebs eingesetzt werden und dort auch gewechselt werden können. Laut Bedienungsanleitung sind diese dafür ausgelegt das Gerät für ein Jahr zu betreiben.

Der Türschlossantrieb ist in der Abbildung 21 dargestellt. Auf der Vorderseite des Geräts befindet sich im oberen Bereich ein Handrad, das eine manuelle Ver- oder Entriegelung des Schlosses ermöglicht. Darunter befinden sich zwei Knöpfe zum Öffnen und Schließen der Tür sowie eine Kontroll-LED, die laut Bedienungsanleitung anzeigt, ob ein Befehl erfolgreich oder nicht erfolgreich ausgeführt wurde, ob das Gerät gerade in den Werkszustand zurückgesetzt wird oder ob die Batteriespannung gering ist.

Abbildung 21: Eqiva Türschlossantrieb

Abbildung 22: App des Equiva Türschlossantriebs

Um den Türschlossantrieb per mobilem Endgerät zu bedienen, kann die App „Eqiva Lock“ (Android, Version 1.3.0) verwendet werden. Dem Türschlossantrieb sind zwei vom Hersteller als „Key Card“ bezeichnete Karten mit einem QR-Code beigelegt, die beim ersten Start der App mit der Kamera des mobilen Endgeräts eingescannt werden sollen. Damit App und Türschlossantrieb miteinander verbunden werden können, muss jedoch neben dem Einscannen des QR-Codes eine der Tasten auf dem Gerät für fünf Sekunden gedrückt gehalten werden, damit dieses in den Pairing-Modus übergeht. Das Gerät wird dann mit der App verbunden, was mit einem kurzen Ton am Gerät bestätigt wird.

In der App nimmt der Benutzer beim Hinzufügen des Türschlossantriebs zunächst einige Einstellungen vor, wie beispielsweise die Drehrichtung zum Öffnen oder Schließen, die Schlüsselposition oder wie oft der Schlüssel zum Verriegeln gedreht werden soll. Diese Einstellungen sind danach auch noch änderbar. Auf der Übersichtsseite der App ist ein Bild eines Hauses zu sehen und entsprechende Buttons zum Entsperren oder Verriegeln sowie Öffnen der Tür. Beim Starten der App wird der aktuelle Zustand der Tür angezeigt und der Benutzer kann diese mit einem Klick entsprechend steuern. Außerdem sind weitere Optionen möglich, wie das Einsehen eines Logs, das neben den Steuerungsbefehlen der App auch anzeigt wann das Schloss manuell oder über die Knöpfe am Antrieb geöffnet oder geschlossen wurde.

Außerdem wird der Benutzer direkt beim Start durch eine Meldung darauf hingewiesen, dass eine neue Firmware-Version für den Türschlossantrieb zur Verfügung steht und gefragt, ob diese installiert werden soll. Ein Firmware-Update wird jedoch zunächst nicht durchgeführt.

6.3.1 Sicherheitsanalyse

Zu Beginn der Analyse werden - wie zuvor auch - zunächst allgemeine Informationen gesammelt. Der Türschlossantrieb verwendet Bluetooth Low Energy und sendet in einem Intervall von jeweils einer Sekunde durchgehend Advertising-Pakete aus, sofern er mit Strom versorgt ist. Die erste Übersicht über die Services und Handles mit „BLEAH“ ergab, dass es insgesamt fünf Handles gibt, die einen Schreibzugriff ermöglichen, einige Weitere einen Lesezugriff. Dabei handelt es sich um die Handles *0x0141* (mit der Bedeutung einer Adresse zum Wiederverbinden), *0x0411*, *0x0421*, *0xFF02* und *0xFF05*.

Beim Mitschneiden der Kommunikation zwischen App und Türschlossantrieb wird deutlich, dass zumindest nicht die von BLE vorgesehene Pairing-Funktionalitäten und anschließende Verschlüsselung verwendet wird. Beim Beginn der Verbindung mit der App wird kein Encryption-Request gesendet, sodass die Kommunikation nicht die vorgesehene Verschlüsselung von BLE verwendet.

Nach dem Verbinden mit der App werden zunächst verschiedene Handles und Attribute abgefragt und in der App wird der aktuelle Zustand des Türschlosses angezeigt (geöffnet oder geschlossen). Danach können durch Drücken der entsprechenden Tasten Befehle z.B. zum Öffnen der Tür gesendet werden. Diese Befehle sehen beim Mitschneiden der Kommunikation wie folgt aus. Es wird zunächst ein Write-Request (also ein Schreibbefehl, der eine Bestätigung vom Gerät erwartet) auf das Handle *0x0411* abgesendet. Auf diesen folgt nach der Bestätigung zusätzlich eine Benachrichtigung an die App aus dem Handle *0x0421*. Nachdem der Befehl vom Türschlossantrieb fertig ausgeführt wurde, sendet das Gerät eine weitere Benachrichtigung aus selbigem Handle und die App sendet erneut einen Write-Request, der anschließend bestätigt wird. Diese Vorgehensweise geschieht sowohl beim Befehl zum Öffnen, als auch beim Befehl zum Schließen.

Beim ersten Verbinden mit der App wird zwar der mitgelieferte QR-Code eingescannt, jedoch findet laut Mitschnitt der Kommunikation kein BLE-Pairing statt. Da die Kommunikation nicht verschlüsselt erfolgt und zuvor kein Pairing durchgeführt wird, werden auch

keine Schlüssel nach BLE-Standard ausgetauscht, die mitgeschnitten werden könnten. Es ist zudem auch kein Downgrade des Verfahrens zum Pairing erforderlich, da kein Pairing durchgeführt wird.

Bei der Analyse der einzelnen aufgenommenen Befehle zum Öffnen und Schließen der Tür fällt auf, dass diese einen ähnlichen Aufbau aufweisen. Alle Befehle haben eine Länge von 16 Byte und beginnen mit „80 87“. Die Write-Requests, nachdem das Gerät den Befehl ausgeführt hat, beginnen mit „80 82“. Sämtliche Benachrichtigungen des Türschlossantriebs nach einem vorherigen Write-Requests beginnen mit „80 83“ und sind ebenfalls 16 Byte lang, während die Benachrichtigungen nach dem erfolgreichen Ausführen eines Befehls mit „80 05“ gefolgt von 14 Byte Nullen. Es ist jedoch auch erkennbar, dass bis auf die ersten beiden Bytes der Schreibbefehle und dem 11. Byte, das bei allen aufgenommenen Befehlen „00“ ist, keine weiteren Bytes gleich bleiben. Es ist also bei mehrfachem Ausführen eines Öffnen- oder Schließen-Befehls keiner dieser Befehle gleich. Gleiches gilt für die Benachrichtigungen (bis auf Ausnahme der oben genannten Benachrichtigungen mit 14 Byte Nullen).

Somit gibt es zwar keinen Pairing-Prozess mit anschließender Verschlüsselung wie sie im Standard von Bluetooth Low Energy vorgesehen ist, sondern eine eigene Vorgehensweise Daten geheim zu halten und zu variieren. Dabei sind die ersten beiden Bytes der Nutzdaten eines BLE-Befehls konstant und dahinter folgen immer 8 Byte an Daten, die für jeden Befehl unterschiedlich sind, gefolgt von einem Byte mit dem Wert null (11. Byte), sowie weitere 5 Byte an zufällig aussehenden Daten. Außerdem ist beim Betrachten in chronologischer Reihenfolge der Befehle, die von der App aus gesendet wurden, auffallend, dass das Byte nach dem 11. Byte augenscheinlich ein Counter ist, der mit jeder Nachricht um *1* erhöht wird.

Was genau für ein Verfahren für die Bytes 3 bis 10 eingesetzt wird, die vermutlich die eigentlichen Nutzdaten sind und ob es sich um eine wirkliche Verschlüsselung handelt oder lediglich ein Vertauschen von Bytes oder etwas ähnliches durchgeführt wird, kann anhand der bloßen Betrachtung der Befehle nicht geklärt werden. Zudem ist auch nicht klar, um was es sich bei den letzten 4 Byte handelt.

Um herauszufinden was die beiden augenscheinlich verschlüsselten Teile des Befehls sind, wird die Android-App dekompiliert. Während es bei der App des Tapplock One in 6.2.1 ohne Probleme möglich war, die App zu dekompilieren und diese in für Menschen lesbarem Code gut zu analysieren war, ist dies bei der App des Eqiva Türschlossantriebs nicht so einfach möglich. Entweder auf Grund von Problemen beim Dekompilieren der App oder aber absichtlich durch den Hersteller hat ein Großteil der Variablen-, Klassen-, Funktions- und Dateinamen der dekompilierten App nicht eine inhaltlich sinnvolle Namensgebung, sondern besteht lediglich aus einer zufällig erscheinenden Anreihung von einzelnen Buchstaben und Zahlen. Dadurch wird sowohl allein schon die Suche entsprechender Stellen zum Zusammensetzen des BLE-Befehls zum Öffnen oder Schließen der Tür deutlich erschwert, als auch die Analyse dieser im Code.

Die Befehle zum Aufschließen, Abschließen und Öffnen der Tür sind nach Analyse des Codes immer gleich aufgebaut. Jeder Befehl ist 16 Byte lang, wovon das erste Byte die Länge des Befehls angibt (in diesem Fall also 128 Bit = 16 Byte = 80 hexadezimal) - was bereits zuvor als statischer Wert bei den aufgenommenen Befehlen identifiziert werden konnte. Das zweite Byte ist der Typ des Befehls, der bei den drei genannten Befehlen jeweils der gleiche ist („COMMAND“) mit einem festen Wert von 87 (hexadezimal), sodass dieser ebenfalls zuvor als statisch identifiziert wurde. Die anderen Befehle, bei denen dieser Wert abweicht, sind vom Typ Status-Requests (82), Status-Info (83) und Status-Changed-Notification (05).

Nach den ersten beiden Bytes folgt ein 8 Byte langer verschlüsselter Teil der eigentlichen Nutzdaten. Im Fall des Verschließens der Tür handelt es sich bei dem eigentlichen Befehl um den Wert *0*, beim Aufschließen um den Wert *1* und das Öffnen geschieht mit dem Wert *2*.

Der jeweilige Wert wird per Exclusive-Or (XOR) mit einem verschlüsselten einmalig gültigen Wert (Nonce) verbunden und dadurch ergeben sich die Nutzdaten. Für die Verschlüsselung wird AES im Modus ECB verwendet, wodurch aus gleichen Befehlen immer gleiche verschlüsselte Befehle resultieren. Jedoch fließt in jeden Befehl auch der aktuelle Nachrichten-Counter mit ein, der mit jeder Nachricht erhöht wird. Somit ist der verschlüsselte Teil bei jeder Nachricht - auch wenn sie den gleichen Inhalt besitzt - anders.

Als Nonce wird ein 8 Byte langer Wert verwendet, der vom Hersteller als „Session-OpenNonce“ bezeichnet wird und beiden Kommunikationspartnern zuvor bekannt ist, da er zu Beginn der Verbindung unverschlüsselt als Write-Request von der App an den Türschlossantrieb gesendet wird.

Der Schlüssel für die Verschlüsselung mit AES wird beim initialen Verbinden mit einer Länge von 16 Byte zufällig generiert und im Befehl „Pairing Request“ an den Türschlossantrieb gesendet. Dieser Befehl ist mit dem auf der „Key Card“ angegebenen Schlüssel verschlüsselt. Selbst beim Mitschneiden dieses initialen Verbindungsvorgangs und der verschlüsselten Übertragung des Schlüssels, ist also eine Kenntnis des Schlüssels auf der mitgelieferten „Key Card“ notwendig, um den Schlüssel entschlüsseln zu können.

Hinter den Nutzdaten wird ein 2 Byte langer Nachrichten-Counter unverschlüsselt angehängt. Die zunächst als statisch vermuteten Nullen im 11. Byte gehören also ebenfalls zu diesem Counter, wie das darauf folgende auch zuvor als Counter identifizierte Byte. Der Counter wird mit jeder Nachricht erhöht. Es fällt zudem auf, dass diese Nonce und auch der Counter bei jeder Verbindung zurückgesetzt werden, d.h. bei jeder Verbindung wird zu Beginn eine Nonce zum Gerät gesendet und der Counter beginnt bei *0*.

Am Ende des Befehls folgt noch ein von den Herstellern als „Security Authentication Value“ bezeichneter Wert. Dieser ist ein Message Integrity Check (MIC) mit einer Länge von 4 Byte, der aus einem Teil der verschlüsselten Nutzdaten und der zuvor dafür verwendeten Nonce berechnet wird. In der Abbildung 23 ist ein beispielhafter Befehl zum Aufschließen der Tür dargestellt.

Abbildung 23: Aufbau eines Befehls zum Aufschließen der Tür

Da alle Befehle immer eine Länge von insgesamt 16 Byte haben, der Typ der Nachricht zwar unverschlüsselt übertragen wird, jedoch Teile der Nachricht verschlüsselt sind, sind - zumindest mit Kenntnissen über die möglichen Nachrichtentypen - begrenzt Rückschlüsse auf die Nachrichteninhalte möglich, zumindest in der Form um welchen Befehlstyp es sich handelt („CMD“ in der Grafik), jedoch nicht auf den vollständigen Inhalt (Nutzdaten).

Es ist zwar möglich aufgenommene Befehle systematisch zu verändern, jedoch können in diesem Fall nicht einfach einzelne Werte angepasst werden, um das Schloss beispielsweise aufzuschließen, da sowohl die Nutzdaten als auch die Integritätsprüfung am Ende des Befehls entsprechend neu berechnet werden müssten. Dafür werden sowohl die aktuelle Nonce benötigt, die jedoch zu Beginn der Verbindung ausgetauscht wird (mit den Befehlen „Connection Request“ und „Connection Info“), als auch der gültige Schlüssel, der nur bei der initialen Verbindung verschlüsselt mit dem Schlüssel der „Key Card“ ausgetauscht wird und somit nicht bekannt ist. Beim Einspielen fehlerhafter Nachrichten wird eine Benachrichtigung „Answer without Security“ zurückgegeben und die Nachricht verworfen. Es wurden mehrere Nachrichten mit zufällig generierten Nutzdaten und zufälligem Message Integrity Check eingespielt. Die Verbindung wurde jedoch nach zehn fehlerhaften Nachrichten in Folge vom Türschloss-

antrieb beendet. Geöffnet werden konnte die Tür dadurch nicht, da ein korrekter Befehl nicht zufällig getroffen wurde.

Die vorhandene Integritätsprüfung am Ende der Befehle zielt darauf ab Manipulationen an den Nutzdaten aufzudecken. Da in dieser ebenfalls eine Kenntnis über den Schlüssel notwendig ist, kann diese ohne den Schlüssel ebenfalls nicht erfolgreich manipuliert werden. Hier gibt es also zwei denkbare Möglichkeiten eine Nachricht zu manipulieren. Eine Möglichkeit wäre es, den Schlüssel zu „erraten“ (Brute Force Angriff), da die Nonce bekannt ist und diese bei Aufnehmen eines Befehls verschlüsselt wird und z.B. mit dem Befehl zum Aufschließen per XOR verknüpft wird. Es wäre also möglich - sofern der aufgenommene Befehl bekannt ist (Aufschließen, Abschließen oder Öffnen) - die verschlüsselte Nonce zu erhalten und die unverschlüsselte Nonce so oft mit einem generierten Schlüssel zu verschlüsseln, bis die verschlüsselte Nonce daraus resultiert. Dies ist also auch ohne Kommunikation mit dem Gerät („offline“) möglich. Der generierte Schlüssel könnte dabei immer erhöht werden oder per Zufallsgenerator erstellt werden (ohne, dass Schlüssel mehrfach vorkommen). Mit diesem könnten dann in Zukunft alle Nachrichten manipuliert werden, da alle Berechnungen selbst durchgeführt werden könnten. Bei einem 128 Bit großen Schlüssel, wie er in diesem Fall verwendet wird, liegt die Anzahl der Kombinationen bei $2^{128} \approx 3,4 \cdot 10^{38}$, von denen eine der gültige AES-Schlüssel ist. Die zeitliche Dauer für den AES-Algorithmus in verschiedenen Modi, u.a. auch der hier verwendete ECB-Modus, wurde in in der Vergangenheit untersucht. Demnach benötigt beispielsweise ein Prozessor vom Typ Intel Core i7 $0,371$ Sekunden um 100 MB Daten zu verschlüsseln [Uskov Byerly Heinemann 2016, S.98f.]. Man würde also die 8 Byte große Nonce aus einer aufgenommenen Kommunikation mit einem zufälligen Schlüssel verschlüsseln und prüfen, ob das Resultat mit der verschlüsselten Nonce in der Nachricht übereinstimmt. Ist dies der Fall, ist der Schlüssel gefunden. Im für den Angreifer schlechtesten Fall, dass der Schlüssel der letzte ist, der ausprobiert wird, liegt die Dauer für einen solchen Brute Force Angriff bei $3,05 \cdot 10^{23}$ Jahren. Selbst bei Aktivieren eines Modus im Prozessor mit sogenannten AES New Instructions (AES-NI), die die Berechnung von AES beschleunigen, dauert ein Brute Force Angriff noch $2 \cdot 10^{23}$ Jahre, sodass es sehr ineffizient ist den Schlüssel auf diese Art und Weise zu „erraten“.

Die andere Möglichkeit besteht darin, die Nachrichtenteile, für die ein Schlüssel notwendig ist, per Brute Force zu „erraten“. Um also einen gültigen Befehl zum Aufschließen der Tür zu finden, ist es notwendig bei 8 Byte (64 Bit) Nutzdaten und 4 Byte (32 Bit) Integritätsprüfung die richtige Kombination zu finden. Es gibt also $2^{96} \approx 7,9 \cdot 10^{28}$ Kombinationen, von denen drei (Aufschließen, Abschließen, Öffnen) zu einem gültigen Befehl führen. Da in diesem Fall die Berechnungen und die Überprüfung, ob der richtige Befehl gefunden wurde, nicht offline durchgeführt werden können und das Abwarten einer Antwort des Türschlossantriebs notwendig ist, um dies herauszufinden, ist die zeitliche Dauer nur ungenau zu berechnen, da die Antwortdauer variiert. Bei Absenden des Befehls als Write-Request sendet das Gerät innerhalb der nächsten durchschnittlich etwa 100 Millisekunden sowohl einen Write-Response als auch eine Antwort in Form einer Benachrichtigung, ob der Befehl fehlerhaft war. Geht man von diesen 100 Millisekunden aus, würde es $2^{96} \cdot 0,1 \approx 7,9 \cdot 10^{27}$ Sekunden $= 2,5 \cdot 10^{20}$ Jahre dauern, um alle möglichen Kombinationen durchzuprobieren. Auch dieser Ansatz ist also nicht praktikabel.

Der Türschlossantrieb wird zwar mit Batterien betrieben und diese könnten somit zu einem bestimmten Zeitpunkt leer werden, jedoch ist es dennoch möglich das Schloss manuell ab- oder aufzuschließen und die Tür auch bei einem Ausfalls des Antriebs zu steuern. Durch diese Möglichkeit der manuellen Steuerung ist also keine Notwendigkeit gegeben, dass sich das Schloss vor dem vollständigen Leeren der Batterien öffnet, um z.B. ein Verlassen der Wohnung im Notfall zu ermöglichen. Nach Bedienungsanleitung des Hersteller ist die einzige Variante, in der sich das Schloss ohne entsprechende Bedienung in der App öffnet, das

Zurücksetzen des Türschlossantriebs auf den Werkszustand. Um dieses Zurücksetzen auszulösen ist jedoch zum einen das Herausnehmen und Wiedereinsetzen der Batterien und zum anderen mehrere Tastendrücke auf dem Gerät notwendig. Über reine Bluetooth Low Energy Befehle über die App ist dies nicht möglich.

Sicherheitskritische Variablen wie der Wert des Nachrichten-Counters oder der Nonce werden ohnehin bei jeder neuen Verbindung neu gesetzt. Der Schlüssel für die Verschlüsselung bleibt auch nach dem Neustart des Geräts erhalten und es muss kein erneutes Pairing (Einscannen der „Key Card“ etc.) durchgeführt werden. Der letzte Zustand des Geräts bleibt also erhalten. Einzig die Position des Antriebs muss beim Neustart einmalig am Gerät neu eingestellt werden, was durch Drücken des Knopfes zum Öffnen oder Schließen geschieht, damit das Schloss einen definierten Zustand besitzt (da das Schloss ohne Strom manuelles Ver- oder Entriegeln ggf. nicht mitbekommen haben könnte). Somit gibt es keine weiteren sicherheitsrelevanten Konsequenzen durch einen Angriff auf die Stromversorgung des Türschlossantriebs.

Beim Anlegen der initialen Verbindung mit dem Türschlossantrieb wird im Gerät ein Benutzer mit einer Benutzer-ID angelegt. Es sind auch mehrere Benutzer mit unterschiedlichen Endgeräten möglich. Über diese ID wird der jeweilige Benutzer identifiziert, indem die ID von der App im „Connection Request“ Befehl mit angegeben wird. Diese könnte zwar relativ einfach manipuliert werden, da sie unverschlüsselt übertragen wird, jedoch gibt es zusätzlich Authentizität durch die Kenntnis des gültigen Schlüssels, mit dem alle Befehle verschlüsselt werden. Die Kombination aus Benutzer-ID und der Möglichkeit korrekt zu ver- und entschlüsseln ermöglicht einem Benutzer das Steuern des Türschlossantriebs. Der Schlüssel ist nur erhaltbar, indem der Benutzer im Besitz eines gültigen Schlüssels auf einer mitgelieferten „Key Card“ ist, die er nicht für Dritte zugänglich machen sollte.

Beim Öffnen der App und Verbinden mit dem Türschlossantrieb wird durch den Befehl „Connection Info“ die aktuelle Versionsnummer des Geräts mitgeteilt (Version 1.6) und der Benutzer wird informiert, wenn es ein Firmware-Update gibt. Er kann dieses dann bestätigen und starten. Wegen einer möglichen späteren Verwendung zu Schulungszwecken soll jedoch kein Firmware-Update durchgeführt werden. Auf Grund nicht zur Verfügung stehenden alten Firmware-Updates kann nicht überprüft werden, ob es möglich ist veraltete Updates einzuspielen (Downgrades).

Durch den Einsatz von Nonce, Counter und Verschlüsselung handelt es sich bei den gesendeten Befehlen zum Auf- und Zuschließen der Tür nicht um statische Befehle. Alle Befehle unterscheiden sich und sind dadurch vor Wiedereinspielen alter Befehle geschützt.

Die Abbildung 24 zeigt die Vorgehensweise beim Verbinden der App mit dem Gerät, nachdem beide bereits zuvor einen gemeinsamen Schlüssel ausgetauscht haben, der mit dem auf der „Key Card“ angegebenen Schlüssel (in Form eines QR-Code) verschlüsselt wurde.

Beim Start der Verbindung sendet die App zunächst einen in der App als „Connection Request“ bezeichneten Befehl, der die ID des Benutzers (in Form eines Integer-Wertes mit einem Byte Größe) und eine zufällig generierte Nonce enthält. Darauf antwortet der Türschlossantrieb mit einer „Connection Info“, die eine Nonce des Geräts beinhaltet und weitere Werte wie die Versionsnummer der Firmware. Es ist zwar möglich die gleiche Nonce im Connection Request zu senden wie sie z.B. in aufgenommenen Befehlen von der App gesetzt wurde, jedoch ist die Geräte-Nonce davon unabhängig, sodass eine andere Geräte-Nonce zurückgegeben wird als im aufgenommenen Befehl. Es können Befehle also nicht wiederverwendet werden, da die Nonce des Geräts in den Befehlen zum Zu- und Aufschließen verschlüsselt wird. Der hierfür verwendete Schlüssel ist jedoch nicht bekannt. Sofern das Gerät auf eine Nonce im „Connection Request“ deterministisch antworten würde, wäre es möglich gewesen auch ohne Kenntnis über den Schlüssel veraltete Nachrichten einzuspielen, die genau die daraus resultierende Nonce verwendeten.

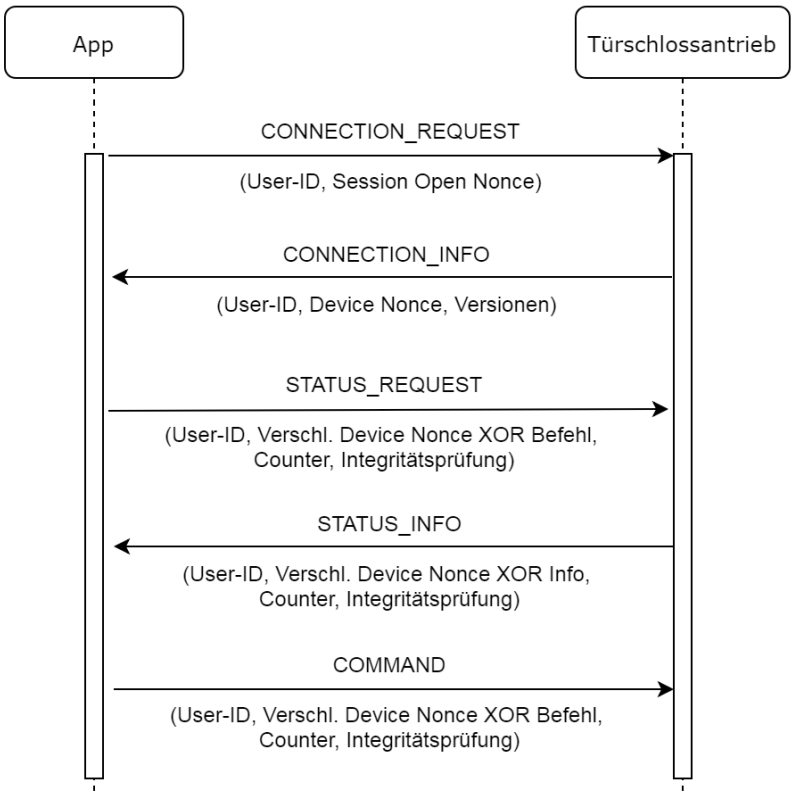

Abbildung 24: Verbindungsaufbau zwischen App und Equiva Türschlossantrieb

Beim Einspielen zuvor aufgenommener Befehl mit einer nicht mehr aktuellen Nonce wird die Ausführung des Befehls vom Türschlossantrieb verworfen und eine Benachrichtigung vom Gerät zurückgegeben, dass der Befehl nicht korrekt ist (Befehl „Answer without Security“) und in diesem Fall würde in der App ein neues Pairing durchgeführt, wofür an dieser Stelle jedoch der gültige Schlüssel auf der „Key Card“ fehlt. Der Türschlossantrieb ist also gegen Wiedereinspielen von Befehlen geschützt.

Durch Fälschen oder Klonen von Geräten und deren Merkmalen ist es nicht möglich den Türschlossantrieb zu steuern, da auch nach Fälschen der Benutzer-ID noch immer der Schlüssel notwendig ist, um den Nutzdatenteil der Nachricht zu verschlüsseln. Es reicht also nicht aus Benutzer-ID oder MAC-Adresse zu fälschen, um an Daten zu gelangen oder das Gerät steuern zu können.

6.3.2 Ergebnis

In der folgenden Tabelle 3 werden die Ergebnisse aus der Sicherheitsanalyse zusammengestellt.

Tabelle 3: Zusammenfassung Analyse Eqiva Türschlossantrieb

Funkprotokoll	Bluetooth Low Energy (BLE)
Verbindungsaufbau	Immer möglich bei Stromversorgung
Stromversorgung	Batterien (wechselbar)

Verschlüsselung	AES-128 (nur bestimmte Befehlsteile)
Schlüsselaustausch	Verschlüsselte Übertragung (mit Schlüssel aus QR-Code auf „Key Card“)
Nachrichtenlänge/-aussehen	Begrenzt Rückschlüsse auf Inhalt möglich
Befehle wiedereinspielbar	Nein
Befehle veränderbar	Nein
Fehlerhafte Nachrichten	Werden ignoriert, Fehlermeldung
Integritätsprüfungen	MIC, CRC
Verhalten nach Neustart	Letzte Einstellungen, einmalig Antriebsposition setzen
Authentifikation	Benutzer-ID und Kenntnis über Schlüssel
Firmware Up-/Downgrades	Updates manuell möglich, Downgrades nicht geprüft

Insgesamt ist festzustellen, dass es durch den Einsatz von Verschlüsselung und Integritätsprüfung für Angreifer schwer ist, den Türschlossantrieb zu steuern. Alle Nachrichten variieren bei jedem Absenden, wodurch ein Wiedereinspielen von aufgenommenen Nachrichten nicht erfolgreich möglich ist. Der Schlüssel, der für die Kommunikation verwendet wird, ist zufällig generiert und wird mit einem Schlüssel, den der Benutzer auf einer mitgelieferten „Key Card“ per QR-Code einscannen kann, verschlüsselt übertragen. Dadurch ist er auch vor dem Mitschneiden der initialen Verbindung zwischen App und Gerät geschützt. Ein Angreifer benötigt den Schlüssel, um die Kommunikation zu manipulieren oder eigene Befehle auszusenden. Alle fehlerhaften Befehle werden nicht ausgeführt.

Der Türschlossantrieb ist also aus Sicht des Funkprotokolls nur so lange gesichert wie der Schlüssel geheim bleibt und dem Angreifer nicht zur Verfügung steht. Gleiches gilt für den mitgelieferten QR-Code auf der „Key Card“.

Durch die Möglichkeit von Firmware-Updates ist der Hersteller zudem in der Lage Änderungen an der Firmware des Geräts vorzunehmen, was beispielsweise genutzt werden kann, um Sicherheitslücken zu schließen, sofern in Zukunft solche entdeckt werden.

6.4 Basetech Raumthermostat

Ein weiteres Gerät zur Steuerung von Heizungen ist das Basetech Raumthermostat. Dabei handelt es sich um ein per Bluetooth Low Energy steuerbares Gerät, das an eine Wand gehangen werden und an eine Heizung angeschlossen werden kann. Auf der vorderen Seite des Thermostats befindet sich ein Display, das im Betriebszustand die aktuellen Einstellungen wie aktuell eingestellte Temperatur, aktuelle Raumtemperatur oder Uhrzeit anzeigt. Darunter befinden sich insgesamt sechs Knöpfe, die eine manuelle Einstellung am Gerät selbst ermöglichen. Je nach getätigten Einstellungen gibt das Thermostat Signale über die entsprechenden Anschlüsse an eine angeschlossene Heizung oder Kühlung weiter, sodass diese darauf reagieren kann. Das Raumthermostat ist auch über eine Smartphone-App steuerbar.

In der Abbildung 25 ist das Raumthermostat abgebildet, die Abbildung 26 zeigt die dazugehörige App „Blue Stat“ (Android, Version 10.0) zum Steuern des Thermostats.

Nach dem Einsetzen der Batterien in das Thermostat ist dieses einsatzbereit. Beim Starten der App wird das entsprechende Thermostat direkt gefunden und als Vorschlag zum Hinzu-

Abbildung 25: Basetech Raumthermostat

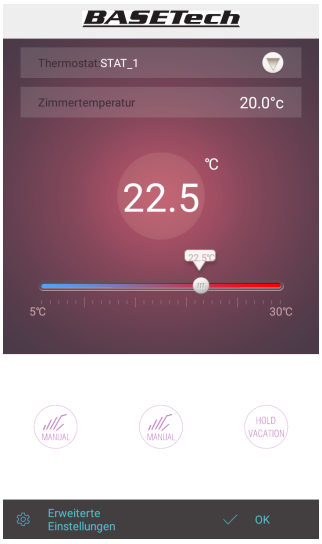

Abbildung 26: App des Basetech Raumthermostats

fügen angezeigt. Nach dem Hinzufügen in der App ist dieses steuerbar. Der Benutzer kann beispielsweise einfache Einstellungen vornehmen, wie das Setzen der gewünschten Temperatur durch einen Klick auf einer Temperaturskala (siehe Abbildung 26), ist aber auch in der Lage komplexere Einstellungen einzugeben, wie zum Beispiel das Anlegen von Programmen (automatische Einstellung der Temperatur je nach Wochentag und/oder Uhrzeit).

Nach dem Tätigen der Einstellungen werden die eingegebenen Temperaturdaten auch auf dem Display des Thermostats angezeigt, bei Änderungen, die die Signalausgänge zur Heizung betreffen, ist auch ein akustisches Geräusch wahrnehmbar, was sich durch Klacken eines Relais äußert.

6.4.1 Sicherheitsanalyse

Auch die Sicherheitsanalyse des Basetech Raumthermostats beginnt damit, zunächst einmal allgemeine Informationen über das Gerät zu erhalten. Nachdem das Gerät eine Stromversorgung durch das Einlegen von Batterien erhält, ist dieses einsatzbereit und kann dann per App über Bluetooth Low Energy gesteuert werden. Dafür sendet das Thermostat mit einem Abstand von 500 Millisekunden Advertising-Pakete aus. Darüber kann die MAC-Adresse erhalten werden, mit der dann wie auch zuvor ein Scan mit „BLEAH“ durchgeführt wird. Dieser schlägt jedoch fehl, da einige Lesezugriffe auf Attribute fehlschlagen, da es nicht ausreichende Authentifikation gibt (Fehlermeldung des Thermostats), um darauf zuzugreifen. „BLEAH“ beendet daraufhin den Suchlauf ohne Anzeige der bisherigen Ergebnisse.

Zumindest einen Teil der Handles für Schreibzugriffe können auch durch Mitschneiden der Kommunikation erlangt werden, indem die Pakete analysiert werden, die von der App ausgesendet werden. Demnach finden Schreibzugriffe (genauer Write-Requests, die eine Antwort des Thermostats erwarten) auf das Handle *0x0025* statt, wodurch die Temperatur am Thermostat eingestellt wird. Dieses Handle ist also relevant, wenn eigene Temperaturbefehle gesendet werden sollen.

Bei der Kommunikation der App mit dem Thermostat fällt zudem auf, dass diese unverschlüsselt abläuft. Zumindest wird kein Pairing nach dem BLE-Standard durchgeführt, der mit einem „Encryption Request“ endet und danach die vorgesehene Verschlüsselung einleitet. Somit ist es möglich, die gesendeten Befehle mitzulesen.

Es wurden verschiedene Temperaturen mit der App gesendet, einige auch mehrfach, um zu überprüfen, ob die resultierenden BLE-Befehle immer gleich sind, wenn es sich um die gleichen Aktionen in der App handelt. Dabei fällt auf, dass in der App zwar beliebig Temperaturen auf der Skala angeklickt werden können, diese werden jedoch erst übernommen, wenn der Benutzer auf den Button „OK“ klickt. Dann wird ein BLE-Befehl gesendet und die Temperatur auch auf dem Display des Thermostats eingestellt und angezeigt. Der Befehl der zum Setzen der Temperatur gesendet wird, besteht aus mehreren Teilbefehlen in Form von Write-Requests auf das Handle *0x0025*.

Insgesamt acht solcher Write-Requests hintereinander werden gesendet, die teilweise inhaltlich immer gleich sind, sich teilweise aber auch unterscheiden, sowohl je nach unterschiedlicher Temperatur, als auch je nach Nachricht. Es konnte festgestellt werden, dass einige Teile der Nachrichten sich bei mehreren Temperatureinstellungen hintereinander ändern, auch wenn die Temperatur immer die gleiche ist. Der erste und letzte Befehl sind jeweils statisch und beinhalten die Zeichen „41 52“ bzw. „63 74“. Der zweite Befehl ist bis auf ein Byte, das sich je nach eingestellter Temperatur ändert, statisch und wird gefolgt von weiteren drei statischen Befehlen. Der sechste und siebte Befehl sind teilweise statisch und teilweise dynamisch mit jedem Befehl und unabhängig von der eingestellten Temperatur.

Um jedoch die genauen Bestandteile des Befehls zum Setzen der Temperatur, insbesondere die dynamischen Teile, herauszufinden, wird die Android-App „Blue Stat“ dekompiliert und der Code im Hinblick auf das Setzen der Temperatur und Absenden des Befehls analysiert. Dabei wird klar, dass ursprünglich ein gesamter Befehl zusammengesetzt wird, der vor dem Absenden in mehrere einzelne Befehle mit einer maximalen Länge von jeweils 19 Byte aufgeteilt wird, zu denen noch ein Byte mit der Länge des Befehls an den Anfang eingefügt wird. So kommen die acht Befehle zu Stande, die nach dem Speichern der Temperatur gesendet werden.

Der Gesamtbefehl setzt sich aus drei Bestandteilen zusammen. Im ersten Teil werden die neu eingegebenen Temperaturdaten übertragen, gefolgt von einer längeren Zeichenfolge, die sich teilweise wiederholt (bei anderen Geräte-Modellen ist diese unterschiedlich lang) und am Ende folgt ein Zeitstempel sowie weitere einstellbare Temperaturwerte. Ein Beispiel für die Befehle ist im Anhang in II.3 dargestellt.

Des Weiteren gilt es herauszufinden, ob bereits Rückschlüsse auf den Inhalt durch Länge oder bestimmte Zeichen der Befehle gemacht werden können. Nachdem der Aufbau der Befehle bekannt ist, sind Rückschlüsse möglich. Es können beispielsweise die eingestellten Temperaturwerte mitgelesen und ausgewertet werden, indem sie aus der Hexadezimaldarstellung in die Dezimaldarstellung umgerechnet werden und halbiert werden.

Außerdem können Befehle mit diesem Wissen systematisch verändert werden. Das bedeutet, dass bestimmte Stellen der Befehle verändert werden können, um z.B. andere Temperaturen am Thermostat einzustellen. Dadurch kann das Thermostat auch durch unautorisierte Dritte gesteuert und Temperaturen verstellt werden. Aufgenommene Befehle oder Teile davon zufällig zu verändern erscheint nicht sinnvoll, da das Thermostat mit der systematischen Veränderung bereits gezielt steuerbar ist. Der Befehlsteil, der den aktuellen Zeitstempel enthält, kann an den aktuellen Zeitstempel angepasst werden. Somit ist es möglich mit einem Python-Script, das die einzelnen Befehle sendet und darin stets den aktuellen Zeitstempel einbindet, das Thermostat auf eine beliebige Temperatur einzustellen.

Das Verhalten beim Einspielen fehlerhafter Nachrichten wurde ebenfalls überprüft. Es wurden fehlerhafte Nachrichten erstellt, in denen Teile eines gültigen Befehls zufällig verändert wurden. Diese Befehle wurden vom Thermostat nicht angenommen und ignoriert. Zudem wurde die Temperatur auf einen ungültigen Wert (49 Grad, maximal 30 Grad in der App möglich) gesetzt. Das Thermostat zeigt auf seinem Display nach diesem fehlerhaften Befehl „Err“ und keine weiteren Daten mehr an. Nach Verbinden mit der App wird zwar

die aktuell eingestellte Temperatur von 49 Grad vom Thermostat eingelesen und angezeigt, obwohl diese eigentlich außerhalb der möglichen Werte in der Skala liegt, es sind jedoch auch gültige Befehle danach nicht mehr möglich. Der Benutzer ist also nicht mehr in der Lage das Thermostat zu steuern, nachdem es in den Fehlerstatus gebracht wurde. Ein Drücken von Tasten auf dem Thermostat zeigt zwar kurzzeitig wieder den aktuellen Zustand wie üblich an, nach einigen Sekunden wird jedoch erneut „Err“ angezeigt. Hilfe für den Benutzer dazu gibt es in der Bedienungsanleitung des Geräts nicht. Das Öffnen des Geräts, die Entnahme der Batterien und ein Neustart des Geräts behebt jedoch den Fehlerstatus und danach ist das Thermostat wieder betriebsbereit.

Eine gesonderte Integritätsprüfung konnte weder in den gesendeten Befehlen, noch im dekompilierten Code der App gefunden werden. Es findet somit also nur die von BLE vorgesehene CRC-Prüfsumme Anwendung, die sich außerhalb der Nutzdaten befindet und nur vor Übertragungsfehlern schützen soll, nicht aber vor Manipulation.

Da das Raumthermostat mit Batterien betrieben wird, können diese nach einer bestimmten Zeit leer werden. Sicherheitsrelevante Konsequenzen, z.B. das Zurücksetzen flüchtiger Variablen wie Counter, gibt es nicht, da keine Counter verwendet werden. Beim Neustart des Geräts bleiben die zuletzt eingestellten Werte erhalten. Allerdings läuft auch ohne Stromversorgung die angezeigte Uhrzeit nicht mit, sodass diese nach dem Neustart ab der letzten Uhrzeit weiterläuft. Ein Angriff auf die Verfügbarkeit könnte so also maximal fest eingegebene Temperaturen nach entsprechenden Uhrzeiten verzögern, je nach Dauer bis zum Austausch der Batterien auch um ganze Stunden, Tage oder Wochen.

Nach der Installation der App benötigt der Benutzer keine Anmeldedaten oder ähnliches um sich mit dem Thermostat zu verbinden. Zwischen App und Gerät wird auch kein Schlüssel ausgehandelt oder ein Pairing-Prozess vollzogen, sodass es dadurch keine Authentizität geben kann. Prinzipiell ist also jeder Benutzer mit entsprechender App in der Nähe in der Lage das Thermostat zu steuern. Generell sind auch Angreifer ohne App in der Lage das Gerät zu steuern, sofern sie Kenntnisse über den Aufbau der Befehle haben. Eine Authentizität von legitimen Benutzern ist also nicht gegeben.

Die Möglichkeit von Updates ist in der Android-App nicht vorgesehen. Auch im dekompilierten Code der App konnte keine solche Funktionalität gefunden werden, sodass davon auszugehen ist, dass Firmware-Updates des Thermostats vom Hersteller nicht vorgesehen sind. Die aktuell installierte Firmware-Version auf dem Thermostat ist zudem auch nicht einsehbar.

Bei den gesendeten Befehlen zum Einstellen der Temperaturen handelt es sich um sich immer ändernde Befehle. Die Ursache dafür konnte nach dem Dekompilieren der App herausgefunden werden. Demnach wird ein Zeitstempel in die Befehle eingefügt, der zunächst vermuten lässt, dass er dafür sorgen soll, dass veraltete Nachrichten nicht mehr ausgeführt werden. Nach dem Aufnehmen von legitimen Nachrichten ist dieser Zeitstempel enthalten und je nachdem wie viel Zeit zwischen Aufnahme und Einspielen der Nachricht vergeht, könnte der Befehl vom Thermostat nicht mehr ausgeführt und verworfen werden. Dieses Verhalten wurde überprüft, konnte so jedoch nicht bestätigt werden. Alle aufgenommenen Nachrichten - auch mit veraltetem Zeitstempel - konnten erfolgreich eingespielt werden und wurden vom Thermostat verarbeitet. Der enthaltene Zeitstempel ist dabei kein Schutz vor Wiedereinspielen von Befehlen, sondern dient lediglich dazu das Datum und die Uhrzeit des Thermostats einzustellen. Nach dem Einspielen veralteter Nachrichten zeigt das Thermostat auch das veraltete Datum und die Uhrzeit aus der Nachricht an.

Somit gibt es entgegen der ersten Vermutungen keinen Schutz vor dem erneuten Einspielen zuvor aufgenommener Nachrichten. Durch fehlende Schutzmechanismen wie Verschlüsselung ist es jedoch auch möglich, den in den Nachrichten enthaltenen Zeitstempel nachträglich zu manipulieren. Es ist dafür die Kenntnis notwendig, welche der Bytes welcher Teil des

Zeitstempels sind und diese können dann an den aktuellen Zeitstempel angepasst werden. Alle übrigen Teile der Nachrichten können aus der aufgenommenen Nachricht übernommen werden. Da es keine Verschlüsselung gibt, befinden sich die Bytes des Zeitstempels immer an der gleichen Stelle. So würde das angezeigte Datum und die Uhrzeit auf dem Thermostat-Display stets aktuell bleiben und es würden dem Benutzer anhand dieses Datums keine Auffälligkeiten in Erscheinung treten.

Da es keine Authentifikation gibt und generell jeder mit Kenntnissen über den Aufbau der Befehle in der Lage ist das Raumthermostat zu steuern, ist es nicht notwendig Geräte zu klonen oder fälschen, die zur Steuerung des Geräts in der Lage sind. Ein Verbindungsaufbau ist mit jedem beliebigen Gerät möglich und alle gesendeten Befehle werden verarbeitet.

6.4.2 Ergebnis

Die Tabelle 4 stellt die Ergebnisse aus der Sicherheitsanalyse zusammen.

Tabelle 4: Zusammenfassung Analyse Basetech Raumthermostat

Funkprotokoll	Bluetooth Low Energy (BLE)
Verbindungsaufbau	Immer möglich bei Stromversorgung
Stromversorgung	Batterien (wechselbar)
Verschlüsselung	Nein
Schlüsselaustausch	Nein
Nachrichtenlänge/-aussehen	Rückschlüsse auf Inhalt möglich
Befehle wiedereinspielbar	Ja
Befehle veränderbar	Ja, vollständig nachstellbar
Fehlerhafte Nachrichten	Werden ignoriert, bei ungültigen Temperaturwerten Fehlerstatus (Neustart notwendig)
Integritätsprüfungen	CRC
Verhalten nach Neustart	Letzter Zustand, Uhrzeit nicht mehr aktuell
Authentifikation	Nein
Firmware Up-/Downgrades	Keine Updates möglich

Die Tabelle 4 zeigt, dass beim Basetech Raumthermostat keine Maßnahmen existieren, um das Gerät vor unbefugter Kontrolle von außen zu schützen. Sämtliche Kommunikation erfolgt unverschlüsselt und somit können Angreifer diese nicht nur mitlesen, sondern auch aufnehmen und erneut einspielen. Die mit jedem Absenden unterschiedlich aussehenden Befehle ließen zunächst vermuten, dass es einen Schutz vor Wiedereinspielen durch einen Zeitstempel gibt, dieser wurde jedoch nur zum Einstellen der Uhrzeit auf dem Thermostat verwendet. Mit Kenntnissen über den Aufbau der Befehle ist es möglich, beliebige Temperaturen einzustellen und weitere Einstellungen am Gerät vorzunehmen. Bei Einspielen fehlerhafter Temperaturwerte ist es möglich das Gerät in einen Fehlerstatus zu bringen, der nur durch einen Neustart (durch Öffnen des Gehäuses sowie Entfernen und Neueinlegen der Batterien) zu beheben ist. Die Befehle sind nicht durch Integritätsprüfungen oder Authenti-

fizierung des Absenders geschützt und somit unentdeckt von jedem Gerät aus manipulierbar. Allerdings kann der legitime Benutzer des Geräts die getätigten Einstellungen durch eigene Einstellungen in der App wieder überschreiben, lediglich im Fehlerstatus ist ein Beheben mit der App nicht mehr möglich. Durch die fehlende Möglichkeit von Updates sind diese Sicherheitsprobleme auch in der Zukunft ohne Austausch des Thermostats nicht zu beheben.

Insgesamt ist das Gerät also als unsicher zu bewerten. Angreifer sind zwar in der Lage das Gerät vollumfänglich zu steuern, allerdings sind die weiteren Auswirkungen hier in der Regel nicht sicherheitskritisch, da es sich - anders als beispielsweise bei einem Vorhängeschloss - nicht um ein Gerät handelt, das Gegenstände oder Räume vor Zutritten schützen oder andere sicherheitskritische Aufgaben übernehmen soll, sondern lediglich um ein Thermostat, das die Raumtemperatur regelt. Denkbar wäre allerdings auch ein Einsatz zum Kühlen von Serverräumen, wobei ein sicherheitskritisches Problem entstehen könnte. Dennoch haben Benutzer des Geräts vermutlich die Erwartung, dass nur sie das Gerät steuern können und nicht jede beliebige Person mit Kenntnissen im Umfeld. Diese Erwartung wird jedoch nicht erfüllt.

6.5 eQ-3 Heizkörperthermostat

Bei dem eQ-3 BLUETOOTH Smart Heizkörperthermostat handelt es sich um ein Gerät zur Steuerung eines Heizkörpers. Das Gerät wird anstelle des Thermostats an der Heizung installiert und die gewünschte Temperatur kann dann über das smarte Heizkörperthermostat eingestellt werden. Möglich ist dabei eine Bedienung des Geräts entweder über die App „calor BT“ (Android, Version 1.1.7) auf einem mobilen Endgerät oder manuell am Thermostat selbst. Die Abbildung 27 zeigt das eQ-3 Heizkörperthermostat. Es wird mit Batterien betrieben, die laut Bedienungsanleitung für eine Stromversorgung über eine Dauer von zwei Jahren ausgelegt sind.

Abbildung 27: eQ-3 Heizkörperthermostat

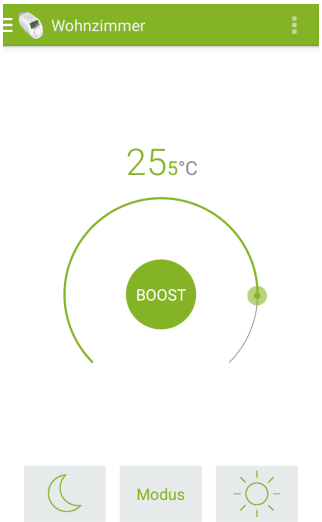

Abbildung 28: App des eQ-3 Heizkörperthermostats

Nach dem Einlegen der Batterien und einer vom Hersteller in der Bedienungsanleitung als Adaptierfahrt bezeichneten Ausrichtung des Reglers an das Ventil der jeweiligen Heizung, befindet sich das Thermostat im „Normalzustand“. Das bedeutet, es zeigt die aktuell eingestellte Temperatur sowie weitere Daten wie Wochentage oder Uhrzeit und dem dazu

eingestellten Programm im Display an der Vorderseite des Thermostats an. Zudem ist standardmäßig Bluetooth eingeschaltet, sodass der Benutzer sich mit der App zum Thermostat verbinden kann. Außerdem sind auch Eingaben am Thermostat selbst möglich. Dafür gibt es zwei Knöpfe jeweils links und rechts sowie ein Stellrad in der Mitte des Geräts.

Verwendet der Benutzer die App des Herstellers zur Steuerung des Geräts, kommuniziert diese per Bluetooth Low Energy mit dem Heizkörperthermostat. Bei der ersten Benutzung der App wird das Thermostat durch Drücken des Stellrades für einige Sekunden in einen Pairing-Modus gebracht, wodurch dem Benutzer auf dem Display eine Nummer bestehend aus vier Ziffern angezeigt wird, die er in der App eingeben muss. Danach sind Thermostat und App gekoppelt und der Benutzer ist in der Lage über einen Regler in der App die gewünschte Temperatur einzustellen und weitere Einstellungen vorzunehmen. Die App mit ihrer Oberfläche zum Einstellen der Temperatur ist in Abbildung 28 dargestellt.

6.5.1 Sicherheitsanalyse

Die Sicherheitsanalyse beginnt auch in diesem Fall mit ersten allgemeinen Informationen über das Gerät und die Kommunikation. Nachdem das Heizkörperthermostat sich im Normalzustand befindet und Bluetooth aktiviert ist, sendet es automatisch und dauerhaft (außer während einer Verbindung) im Abstand von circa einer Sekunde Advertising-Pakete aus, die es ermöglichen von Geräten in der Umgebung gefunden zu werden.

Mit Hilfe von „BLEAH“ können weitere Informationen über das Gerät, z.B. die MAC-Adresse, erlangt werden. Die Übersicht nach dem Abfragen der Services und Handles zeigt, dass es mehrere Handles gibt, die einen Lesezugriff erlauben und ebenfalls mehrere Handles mit Schreibzugriff vorhanden sind. Es kann eine Adresse für das Wiederverbinden geschrieben werden, eine Characteristic mit dem Handle *0x0411*, auf die gleichzeitig auch ein Lesezugriff erfolgen kann, eine Characteristic mit dem Handle *0xFF02* sowie eine Characteristic mit dem Handle *0xFF05*. Hier ist also zunächst noch nicht klar, wo welche Befehle geschrieben werden und welche Bedeutungen die Characteristics haben.

Das Mitschneiden der Kommunikation zwischen App und Thermostat ergibt, dass beide miteinander verschlüsselt kommunizieren. Dies ist daran zu erkennen, dass nach dem Verbinden der Geräte von der App der Befehl „Encryption Request“ gesendet wird, gefolgt von einem „Encryption Response“ des Thermostats sowie ein Befehl zum Starten der Verschlüsselung. Danach werden im TI Packet Sniffer die Datenpakete auch als „Security Enabled“ markiert. Alle verschlüsselten Daten enthalten zum Schutz vor Manipulation der verschlüsselten Nutzdaten zusätzlich noch einen Message Integrity Check (MIC) wie es im BLE-Standard vorgesehen ist.

Die Initiierung der Verschlüsselung wurde bereits im Groben in der Beschreibung und Einrichtung des Geräts beschrieben. Demnach beinhaltet das Thermostat zwar keine Möglichkeit zur Eingabe, sehr wohl aber über ein Display zur Ausgabe eines Nummernwertes. Das mobile Endgerät mit der App verfügt über die Möglichkeit sowohl von Eingaben als auch Ausgaben. Somit ist es möglich - und diese Vorgehensweise findet auch statt -, dass auf dem Thermostat ein zufälliger Wert generiert wird, den der Benutzer nach dem Ablesen auf dem Display des Thermostats in seiner App eingibt und mit diesem Wert kann dann das Bluetooth Pairing-Verfahren „Passkey Entry“ durchgeführt werden, wobei der eingegebene Wert als temporärer Schlüssel verwendet wird. Somit gibt es weder fest einprogrammierte Schlüssel, die gegebenenfalls öffentlich werden könnten (z.B. durch Dekompilieren), noch gibt es eine unverschlüsselte Übertragung von Schlüsseln. Zudem ist ein Man-in-the-Middle-Angriff hier nicht möglich, da der Wert nicht übertragen wird, sondern nur vom Benutzer optisch erfasst und dann in der App eingegeben wird. Der vom Benutzer gelesene und in der App eingegebene Schlüssel sorgt dafür, dass dieser Schlüssel nicht per drahtloser Übertragung

ausgetauscht wird und somit nicht mitgeschnitten werden kann. Ein Angreifer kann also ohne physikalischen Zugriff auf das Thermostat im Moment des Pairings diesen Schlüssel nicht erlangen. Bei allen weiteren Verbindungen muss kein neuer Schlüssel ausgetauscht werden, die Kommunikation kann direkt verschlüsselt erfolgen.

Eine weitere Frage ist die, ob es möglich ist auch ein unsicheres Verfahren als Pairing-Methode zu verwenden oder im Allgemeinen Downgrades auf unsicherere Verfahren möglich sind. Ein Downgrade bei der Initiierung der Verschlüsselung ist insofern möglich, dass das Thermostat auch unverschlüsselte Verbindungen - auch nachdem es zuvor bereits einen Schlüsselaustausch gab - zulässt und keine Verschlüsselung von sich aus fordert. Standardmäßig verwendet die zugehörige App jedoch Verschlüsselung und initiiert diese durch Senden des „Encryption Request“. Kenntnisse über die genauen Befehle um das Thermostat zu steuern, sind jedoch durch Mitlesen der Kommunikation mit der App nicht möglich, da diese Befehle eben verschlüsselt sind.

Standardmäßig wird in Bluetooth Low Energy für die Verschlüsselung der Daten AES-128 verwendet. Somit gibt es keine Klartexte wie Passwörter oder andere sensible Daten, die mitgelesen werden könnten. Befehle sehen aus wie die zufällige Aneinanderreihung von Bytes. Eine Möglichkeit dennoch etwas über den Inhalt der Nachrichten zu erfahren, auch ohne Kenntnis über den Schlüssel zu haben, könnte durch das Analysieren der verschlüsselten Nachrichten bestehen. Es könnten beispielsweise Befehle für bestimmte Temperatureinstellungen unterschiedlich aussehen, d.h. unterschiedliche Längen besitzen oder bestimmte Zeichen häufiger oder seltener beinhalten. Zur Überprüfung dieser Möglichkeit werden per App mehrere gewünschte Temperaturen an das Thermostat gesendet und die Kommunikation mitgeschnitten, um die gesendeten Befehle zu analysieren. Genauer wurden die Temperaturen 21 Grad, 22.5 Grad, 24 Grad, 25 Grad, 26.5 Grad und 27.5 Grad im Regler der App nacheinander ausgewählt.

Dabei konnte festgestellt werden, dass ein Befehl zum Verändern der Temperatur immer aus drei aufeinanderfolgenden BLE-Befehlen besteht. Bei allen eingestellten Temperaturen hatte der erste Befehl eine Länge von 9 Byte und wurde an das Thermostat gesendet, der zweite Befehl war eine Antwort des Thermostats mit einer Länge von 5 Byte und der dritte ebenfalls eine Antwort vom Thermostat mit einer Länge von 13 Byte, jeweils gefolgt von der Integritätsprüfung (MIC). Somit haben die Befehle für unterschiedliche Temperaturen und deren Antworten immer die gleichen Nachrichtenlängen. Die Zeichen der jeweiligen Befehle sehen zudem sehr unterschiedlich aus. Es konnten keine Regelmäßigkeiten oder Abhängigkeiten zu den eingestellten Temperaturen erkannt werden, wie es vom Standard auch beabsichtigt ist. Anhand der Länge und der Zeichen der verschlüsselten Nachrichten sind also ohne gültigen Schlüssel keine Rückschlüsse auf die Inhalte möglich. Es existiert außerdem neben dem Regler für die Temperatureinstellung in der App noch ein Button „Boost“, der dafür sorgt, dass sich das Thermostat sofort vollständig öffnet, um den Raum möglichst schnell zu beheizen. Auch dieser Befehl ist 9 Byte lang und ändert sich fortwährend. Die Antworten des Thermostats darauf sind ebenfalls analog zu den oben genannten Nachrichten als Antwort auf die „gewöhnlichen“ Temperatureinstellungen.

Durch den Einsatz der Verschlüsselung ist es nicht möglich den Aufbau und die Struktur der Nachrichten, die die App zur Steuerung an das Thermostat sendet, zu erfahren. Jeder Befehl sieht unterschiedlich aus, auch wenn es sich dabei inhaltlich immer um den gleichen Befehl handelt.

Aus diesem Grund wurde die App „calor BT“, die zur Steuerung des Thermostats verwendet wird, dekompiliert, um weitere Informationen über die genauen Befehle zu erhalten. Nach dem Dekompilieren ist der Programmcode gut lesbar und es sind keine Maßnahmen getroffen worden, um das Dekompilieren und Lesbarkeit durch Menschen zu erschweren. Demnach können die wichtigsten Befehle zur Steuerung des Thermostats - die zum Setzen

der Temperatur sowie der zum Auslösen der „Boost“-Funktion - herausgefunden werden. Der Befehl zum Einstellen der Temperatur besteht lediglich aus einem festen Byte-Wert (41 hexadezimal) gefolgt von einem Byte des gewünschten Temperaturwertes (multipliziert mit 2, damit auch halbe Temperaturwerte möglich sind), der per Write-Request an das Handle *0x0411* geschrieben wird. Daraufhin folgt eine Antwort des Geräts und eine Benachrichtigung mit dem aktuellen Status, was die drei zuvor aufgenommenen Befehle unterschiedlicher Längen erklärt. Die Befehle sind ebenfalls im Anhang unter II.4 aufgeführt und erklärt. Es handelt sich also um statische Befehle, die nur durch den Einsatz von Verschlüsselung bei jedem Absenden unterschiedlich aussehen. Im Befehl selbst kommen keine Informationen vor, die nur der legitime Absender besitzt.

Standardmäßig wird in Bluetooth Low Energy beim Einsatz von Verschlüsselung auch eine Integritätsprüfung angehängt, die darauf abzielt Manipulationen an der Nachricht nicht unbemerkt zu lassen. Das Einleiten dieser Schutzmaßnahmen - Verschlüsselung und Integritätsprüfung - wird durch die App direkt nach dem Verbinden durchgeführt.

Da jedoch der notwendige Schlüssel nicht bekannt ist, um die nun bekannten Befehle korrekt zu verschlüsseln, wird zunächst versucht eine unverschlüsselte Verbindung zum Thermostat aufzubauen. Bei dieser sendet das Thermostat von sich aus keine Anfrage zur Verschlüsselung und lässt auch Befehle unverschlüsselt zu. Somit wurden die korrekten Befehle unverschlüsselt an das Gerät gesendet. Diese wurden ohne Fehler angenommen und ausgeführt. Es wird also standardmäßig bei der Kommunikation mit der App Verschlüsselung eingesetzt, jedoch werden vom Thermostat auch unverschlüsselte Verbindungen angenommen und darin abgesendete Befehle ausgeführt. Somit ist das Gerät vollständig von Angreifern steuerbar, wenn diese Kenntnisse über den Aufbau und Inhalt der Befehle besitzen. In diesem Fall war es möglich die eingestellte Temperatur auf beliebige Werte zu setzen und die Boost-Funktion zu aktivieren oder zu deaktivieren. Somit war die vollständige Kontrolle des Thermostats möglich ohne im Besitz des gültigen Schlüssels zu sein.

Zudem wird geprüft, was beim Absenden fehlerhafter Nachrichten geschieht. Dafür werden beliebige Nachrichten (z.B. mit dem Inhalt „12 34 56“) per Python-Script an das Thermostat gesendet. Diese werden vom Gerät angenommen, aber es wird mit einer Benachrichtigung mit dem Wert „02 80“ geantwortet und keine (sichtbare) Aktion ausgeführt. Auf dem Display am Thermostat wird ebenfalls kein Fehler angezeigt, sodass davon auszugehen ist, dass falsche Meldungen ignoriert werden. Bei inhaltlich falschen Meldungen ist dieses Verhalten anders. In der App ist es möglich die Temperatur auf einen maximalen Wert von 29.5 Grad einzustellen. Es wurden in der App unmöglich einstellbare Werte (z.B. 80.5 Grad) an das Thermostat gesendet und solche in der App ungültigen Befehle werden angenommen und ausgeführt, sodass sich das Thermostat vollständig aufgedreht und auch die eingestellten (eigentlich gar nicht möglichen) Temperaturen auf dem Display anzeigt.

Bei einem Ausfall des Thermostats auf Grund von leeren Batterien bleibt die zuletzt eingestellte Motoreinstellung (und damit auch die entsprechende Einstellung an der Heizung) erhalten und der Nutzer ist nicht mehr in der Lage seine Heizung zu steuern. Auch eine manuelle Steuerung ist nicht möglich. Beim Neustart nach einem Batteriewechsel erfordert das Thermostat eine Eingabe des aktuellen Datums und der Uhrzeit sowie das Auslösen einer erneuten Adaptierfahrt. Auch die letzte Temperatureinstellung geht verloren und das Gerät beginnt mit einer Standardtemperatur von 17 Grad nach der Adaptierfahrt. Der Schlüssel für die gesicherte Kommunikation mit der App ist jedoch nach wie vor vorhanden und muss nicht neu ausgetauscht werden, sodass ein Neustart des Geräts keine Möglichkeit für Angreifer bietet, um den Schlüssel zu erhalten oder einen neuen Schlüsselaustausch mitschneiden zu können. Bis auf den Schlüssel sind jedoch alle anderen Eigenschaften zurückgesetzt und müssen vom Benutzer neu eingegeben bzw. eingestellt werden, bis das Gerät wieder wie vor dem Neustart einsatzbereit ist.

Die Authentizität des Benutzers wird dadurch gewährleistet, dass es beim Pairing-Prozess die Eingabe einer Nummernfolge gab, die am Thermostat angezeigt wurde. Der daraus berechnete Schlüssel wird in beiden Geräten gespeichert und beim Wiederverbinden können beide Geräte ohne erneuten Schlüsselaustausch verschlüsselt kommunizieren. Dieser Schlüssel ist eine Information, die nur die beiden legitimen Geräte besitzen und authentifiziert somit den Benutzer gegenüber dem anderen Gerät. Weitere Informationen wie Passwörter o.Ä. müssen vom Benutzer nicht angegeben werden. Da das Thermostat jedoch auch unverschlüsselte Verbindungen zulässt und die darin enthaltenen Befehle annimmt, ist in diesem Fall keine Authentizität vorhanden. Eine solche Verbindung ist von jedem Angreifer möglich.

In der App gibt es ebenfalls die Funktion Firmware-Updates durchzuführen. So besteht also die Möglichkeit auch nachträglich noch Sicherheitslücken zu schließen. Dies geschieht jedoch nicht automatisch, sondern der Benutzer muss in einem der Menüs in der App den Punkt „Firmware aktualisieren“ anklicken. Zum Zeitpunkt der Analyse befindet sich das Heizkörperthermostat bereits in der aktuellen Firmware-Version, sodass es nicht möglich ist ein Update durchzuführen. Die genaue Firmware-Version ist in der App nicht ersichtlich. Da App und Thermostat verschlüsselt kommunizieren, kann diese auch beim Abfragen einer neueren Version nicht mitgelesen werden. Es war nicht möglich eine alte Firmware-Version für das Gerät zu finden und es konnte daher nicht überprüft werden, ob es möglich ist eine alte Firmware-Version einzuspielen (ein Downgrade durchzuführen).

Der Einsatz von Verschlüsselung wie er bei der Kommunikation mit der App verwendet wird, ist im Standard von Bluetooth Low Energy so vorgesehen. [Heydon 2013, S.106f.] beschreibt diesen im Detail. Demnach beinhalten die verschlüsselten Nutzdaten (Payload) auch noch einen einmalig gültigen Wert (Nonce), der aus einem Counter sowie einer Richtungsangabe und einem Initialisierungsvektor besteht. Mit jeder nichtleeren Nachricht wird der Counter erhöht, sodass sich die Nachricht insgesamt immer verändert, auch wenn der eigentliche Inhalt gleich bleibt. So kommt es zu Stande, dass bei den Nachrichten des Thermostats keine gleichen Nachrichten beim gleichen Befehl resultieren und Wiedereinspielen von Nachrichten nicht möglich ist.

Ein Klonen oder Fälschen von Gerätemerkmalen hilft an dieser Stelle nicht weiter, da Kenntnisse über den Schlüssel notwendig sind, um verschlüsselte Nachrichten zu senden. Dadurch, dass es jedoch auch möglich ist unverschlüsselt Befehle zu senden, ist es ohnehin für jedes Gerät möglich das Thermostat zu steuern. Somit entfällt dieser Punkt in der Analyse.

6.5.2 Ergebnis

Die Tabelle 5 stellt die Ergebnisse der Sicherheitsanalyse im Folgenden zusammen.

Tabelle 5: Zusammenfassung Analyse eQ-3 Heizkörperthermostat

Funkprotokoll	Bluetooth Low Energy (BLE)
Verbindungsaufbau	Immer möglich bei Stromversorgung
Stromversorgung	Batterien (wechselbar)
Verschlüsselung	AES-128 (nach BLE-Standard)
Schlüsselaustausch	Passkey Entry (nach BLE-Standard)
Nachrichtenlänge/-aussehen	Keine Rückschlüsse auf Inhalt möglich
Befehle wiedereinspielbar	Nein

Befehle veränderbar	Nein, aber vollständig nachstellbar
Fehlerhafte Nachrichten	Werden ignoriert, ungültige Temperaturwerte möglich
Integritätsprüfungen	MIC (nach BLE-Standard), CRC
Verhalten nach Neustart	Einstellungen zurückgesetzt, Schlüssel bleibt erhalten
Authentifikation	Kenntnis über Schlüssel, aber unverschlüsselte Kommunikation wird dennoch akzeptiert
Firmware Up-/Downgrades	Updates manuell möglich, Downgrades nicht geprüft

Die Sicherheitsfunktionen, die im eQ-3 Heizungsthermostat implementiert sind, folgen dem Standard von Bluetooth Low Energy. Es existiert ein Schlüsselaustausch, der gegen Man-in-the-Middle dadurch geschützt ist, dass ein Zufallswert auf dem Display des Thermostats angezeigt wird, den der Benutzer in der App eingibt. Mit diesem Zufallswert wird ein temporärer Schlüssel berechnet, mit dem die weiteren Schlüssel ausgetauscht werden können. Die anschließende Verschlüsselung der Nutzdaten ist zusätzlich durch eine Integritätsprüfung vor Manipulation geschützt. Es ist nicht möglich Nachrichten zu interpretieren oder wiedereinzuspielen, da sie verschlüsselt sind und gleiche Nachrichten nicht gleich verschlüsselt werden.

Allerdings ist es durch Dekompilieren und Analysieren des Programmcodes der App möglich Informationen über die genauen gesendeten Befehle zu erhalten. Da die Verschlüsselung durch die App initiiert wird und das Thermostat von sich aus nicht zwingend eine verschlüsselte Verbindung erfordert, ist es ebenfalls möglich eine ungesicherte Verbindung aufzubauen und Befehle unverschlüsselt zu versenden. Es handelt sich bei den relevanten Befehlen um statische Befehle, die keine weiteren Informationen beim Angreifer voraussetzen, sodass diese einfach nachgestellt werden können. Die unverschlüsselt gesendeten Befehle werden vom Gerät als gültig angesehen und ausgeführt.

Somit ist das Gerät von beliebigen Angreifern vollständig steuerbar und ist als unsicher einzustufen, obwohl die Sicherheitsfunktionen von BLE in der App umgesetzt wurden. Der Hersteller wurde über die oben genannten Erkenntnisse informiert und angefragt, ob diese Sicherheitsproblematik bereits bekannt ist und eventuell in einem zukünftigen Update behoben wird. Bis zum Ende dieser Arbeit wurde die Anfrage zwar an die Entwicklungsabteilung weitergeleitet, jedoch noch nicht abschließend beantwortet.

6.6 iHealth Blutdruckmessgerät

Das iHealth View ist ein Blutdruckmessgerät, das dazu verwendet wird, um den Blutdruck und die Herzfrequenz am Handgelenk eines Menschen zu messen. Dafür befindet sich an der Unterseite des Geräts eine Manschette, die um den Arm gelegt wird. Diese pumpt sich dann mit Luft auf und das Gerät bestimmt für einen bestimmten Zeitraum Blutdruck und Puls der Person.

Das Blutdruckmessgerät besitzt auf der weißen unscheinbaren Oberseite ein Display, das die Ergebnisse der aktuellen Messung und den Batteriestand anzeigt. An der Oberseite des Geräts gibt es einen Knopf „Start/Stop“, der zum An- oder Ausschalten des Geräts verwendet wird sowie zum Einschalten der Bluetooth-Funktion. Diese ist nicht standardmäßig aktiviert, sondern muss gesondert eingeschaltet werden. Eine Stromversorgung erhält das Messgerät durch fest verbaute Akkus, die durch den Benutzer nicht ausgetauscht werden können. Jedoch gibt es an der gegenüberliegenden Seite des Knopfes einen USB-Anschluss, worüber die Akkus mit dem mitgelieferten USB-Kabel wieder aufgeladen werden können.

In der Abbildung 29 ist das Blutdruckmessgerät dargestellt. Daneben befindet sich in Abbildung 30 ein Screenshot der dazugehörigen App „iHealth“ (Android, Version AVT3.7.3), in der beim ersten Start bereits beispielhafte Messdaten aus dem Jahr 2015 eingetragen sind.

Abbildung 29: iHealth Blutdruckmessgerät

Abbildung 30: App des iHealth Blutdruckmessgeräts

Das Gerät speichert die gemessenen Werte und diese können mit der zugehörigen App „iHealth“ abgefragt werden. Die App dient dazu mehrere Messungen hintereinander übersichtlich darzustellen und alle Messungen miteinander vergleichen zu können.

Nach der Installation der App muss zunächst ein Account angelegt werden, der gleichzeitig Zugang zur iHealth-Cloud bietet, in der dann die Daten gespeichert werden. Für die Verbindung zum Blutdruckmessgerät ist keine Eingabe eines Passwortes oder eines PINs erforderlich. Das Gerät wird gefunden und bei Auswählen dieses wird dann die Verbindung hergestellt. Danach können die Daten auf dem Messgerät in der App durch Drücken des Buttons „SYNC“ synchronisiert werden. Dabei werden die Daten der letzten Messungen einmalig übertragen und sind anschließend nicht mehr auf dem Gerät.

Dem Benutzer werden die Messdaten wie systolischer Blutdruck, diastolischer Blutdruck und Puls grafisch aufbereitet. Dies erfolgt entweder über eine Liste, in der alle zuletzt gemessenen Werte mit Datum und Uhrzeit hintereinander angezeigt werden oder aber über verschiedene Diagramme, die eine Vergleichbarkeit der Messergebnisse ermöglichen sollen.

6.6.1 Sicherheitsanalyse

Auch das Blutdruckmessgerät arbeitet mit Bluetooth Low Energy und wird zunächst mit „BLEAH“ untersucht, um erste Informationen über das Gerät zu bekommen. Das Messgerät sendet von sich aus erst einmal keine Advertising-Pakete aus und hat standardmäßig die Bluetooth-Funktion deaktiviert. Erst durch zweisekündiges Drücken der Taste „Start/Stop“ wird diese aktiviert und das Gerät sendet im Abstand von 20 bis 30 Millisekunden fünf Minuten lang Advertising-Pakete aus, sodass das Gerät von der App gefunden werden kann. Darin enthalten ist die MAC-Adresse des Geräts. Es kann außerdem ein Scan mit „BLEAH“ durchgeführt werden, der Informationen über die Handles sowie Schreib- und Leserechte des Messgeräts gibt. Neben einigen lesbaren Eigenschaften wie Hersteller und Name des Geräts, besitzt das Messgerät drei schreibbare Handles *0x0032*, *0x0034* und *0x0065*.

Beim Mitschneiden der Kommunikation fällt auf, dass zwar die Advertising-Pakete des

Blutdruckmessgeräts aufgezeichnet werden können, sobald jedoch eine Verbindung zwischen Messgerät und App zu Stande kommt, ist es nicht mehr möglich die Verbindung mitzuschneiden. Bei der dafür eingesetzten Hard- und Software handelt es sich jedoch nach wie vor um solche zum Mitschneiden von reiner Kommunikation mit Bluetooth Low Energy. Der Hersteller beschreibt seine eingesetzte Technologie zwar mit „Bluetooth V4.1“ und der Betrieb mit einem Akku lässt ebenfalls vermuten, dass aus Gründen der Energiesparsamkeit Bluetooth Low Energy verwendet wird, dies scheint jedoch nicht in der App der Fall zu sein. Die dennoch aufnehmbaren Advertising-Pakete sind auch dadurch zu erklären, dass das klassische Bluetooth und Bluetooth Low Energy auf den gleichen Frequenzen arbeiten, BLE jedoch eine geringere Anzahl von Kanälen verwendet. Da es weder möglich war die Kommunikation mit dem CC2540, noch mit dem BBC micro:bit aufzunehmen und keine Hard- und Software vorliegt, um klassisches Bluetooth aufzunehmen, wird auf dem Androidgerät, das die „iHealth“ App ausführt, in den Entwickleroptionen von Android die Option „Bluetooth HCI Snoop Log“ aktiviert. Mit Auswählen dieser Einstellung wird für jede Bluetooth-Kommunikation eine Logdatei geschrieben, in der die Befehle aufgeführt sind, die gesendet und empfangen werden. So kann überprüft werden, ob App und Messgerät wirklich über Bluetooth Low Energy oder über klassisches Bluetooth kommunizieren.

Die Logdatei ist so formatiert, dass sie mit der Software „Wireshark“ geöffnet werden kann. Darin ist erkennbar, dass es sich bei der Kommunikation mit dem Blutdruckmessgerät tatsächlich nicht um Bluetooth Low Energy, sondern klassisches Bluetooth handelt. Zu erkennen ist dies daran, dass bei einer BLE-Verbindung - für diesen Zweck wurde ebenfalls eine solche Logdatei bei der Kommunikation mit dem Basetech Raumthermostat, das mit BLE kommuniziert, erstellt - ein Verbindungsbefehl gesendet wird, der in Wireshark als „Send LE Create Connection“ bezeichnet wird. Im Fall des Blutdruckmessgeräts wird jedoch ein Verbindungsbefehl „Send Create Connection“ gesendet. Das fehlende „LE“ zeigt hier bereits an, dass es sich nicht um eine Verbindung mit Bluetooth Low Energy handelt, sondern um klassisches Bluetooth. Aus diesem Grund kann die Kommunikation also nicht wie bei den anderen Geräten mit der entsprechenden zuvor vorgestellten Hard- und Software für BLE analysiert werden. Es werden stattdessen die Android-Logdateien für die weitere Analyse verwendet. Da das Blutdruckmessgerät die BLE-Befehle von „BLEAH“ korrekt verarbeitet hat und eine BLE-Verbindung ohne Probleme möglich war, bleibt unklar, wieso der Hersteller bei der Kommunikation mit der App Bluetooth und nicht Bluetooth Low Energy verwendet, wenn beide Geräte offensichtlich in der Lage sind BLE zu verwenden.

Für die Kommunikation mit klassischem Bluetooth kann auch nicht das zuvor immer verwendete Python-Framework „bluepy“ eingesetzt werden, da dieses nur für Bluetooth Low Energy vorgesehen ist. Für das Senden von klassischem Bluetooth wird stattdessen das Frame „pybluez“ verwendet, das ebenfalls mit Python arbeitet und so eigene Scripts ermöglicht, die sowohl eigene Befehle an Bluetooth Geräte senden, als auch jeweilige Antworten verarbeiten können. Ein Pendant zu dem Scan mit „BLEAH“ in BLE bietet für Bluetooth ebenfalls „pybluez“, mit dem ein Suchlauf nach Services auf dem Bluetooth-Gerät durchgeführt werden kann, welcher dann ausgegeben wird. Darin sind beispielsweise der genaue Protokolltyp und ein Port angegeben, was für den Verbindungsaufbau mit dem Gerät erforderlich ist.

Analysiert wird im Folgenden zunächst die Kommunikation zwischen der App und dem Messgerät mittels der Logdatei von Android. In der Logdatei sind zusätzlich weitere Daten enthalten, die zwischen Host und Controller ausgetauscht werden (z.B. Veränderung von Sendeparametern). Interessant sind in diesem Fall jedoch die Nutzdaten (Payload), die zwischen den Geräten ausgetauscht werden. Die MAC-Adressen sind ebenfalls in der Datei aufgelistet und stimmen mit den Ergebnissen aus „BLEAH“ überein.

Bei einem ersten Blick auf die gesendeten und empfangenen Nutzdaten fällt bereits auf, dass alle von der App gesendeten Daten mit „B0“, alle vom Blutdruckmessgerät gesendeten

Daten mit „A0“ beginnen, gefolgt von einer Längenangabe (die jedoch um 1 verringert ist als die wirkliche Länge in Bytes ab dieser Stelle), gefolgt von jeweils einem Byte mit dem Wert „00“ und „A1“. Daraus lässt sich folgern, dass alle Befehle zumindest teilweise unverschlüsselt sind. Der zweite Befehl - ein Befehl vom Messgerät, der als Antwort auf einen Befehl der App gesendet wird - zeigt jedoch, dass die gesamte Nachricht unverschlüsselt ist. Er beinhaltet Informationen zum genauen Gerät wie die Version oder den Gerätetyp (mit der App sind verschiedene Geräte des Herstellers steuerbar).

Das Blutdruckmessgerät besitzt zwar die Möglichkeit eines Displays und der Anzeige einer Ziffernfolge, jedoch gibt es keinen initialen Pairing-Prozess, der diese Möglichkeit nutzt. In den Android-Logs gibt es keine Nachrichten, die einen Pairing-Prozess oder eine Verschlüsselung einleiten.

Bei mehrfachem Verbinden und synchronisieren mit der App fällt außerdem auf, dass die Nachrichten und deren Reihenfolgen immer gleich sind. Die zweite Nachricht ist beispielsweise immer eine Antwort des Geräts mit Informationen zum Blutdruckmessgerät wie Gerätetyp und Version. Wenn eine neue Blutdruckmessung durchgeführt wurde, wird diese in der App nach dem Synchronisieren hinzugefügt und angezeigt. Wenn keine neue Messung vorhanden ist, wird eine Meldung angezeigt, dass keine neuen Daten vorliegen. Auch dieses Verhalten ist in den Bluetooth-Nachrichten wiederzuerkennen. Die letzten Befehle der App sind immer die gleichen, allerdings unterscheidet sich die Antwort des Messgeräts je nachdem ob neue Daten vorliegen oder nicht. Es wird die Anzahl der neuen Datensätze mit einem Byte-Wert übertragen und wenn dieser mindestens *1* ist, folgt ein weiterer Befehl der App, auf den eine etwas längere Antwort des Messgeräts folgt.

Dabei handelt es sich um die Messdaten, die an die App übertragen werden. Der Befehl beinhaltet neben einem festen ersten Teil auch einen Zeitstempel der Messung, sowie den gemessenen systolischen Blutdruck, diastolischen Blutdruck und den Puls. Der systolische Druck wird dabei als Differenz zum diastolischen Blutdruck angegeben.

Dadurch, dass die Antworten ebenfalls unverschlüsselt erfolgen, sind Rückschlüsse auf den Inhalt (nämlich alle gesendeten Messdaten) der Nachrichten durch Angreifer möglich und dieser kann - mit vorhandener Hardware zum Mitschneiden der klassischen Bluetooth-Kommunikation - die gemessenen Werte mitlesen.

Allerdings gibt es auch Befehle, die nicht statisch sind. Diese beginnen zwar ebenfalls immer mit der gleichen Struktur, allerdings beinhalten sie Zeichenketten, die sich bei jeder Verbindung unterscheiden.

Für genauere Informationen über den Aufbau der Befehle wird die App dekompiliert und der Programmcode genauer auf die Bestandteile der Befehle analysiert. Demnach handelt es sich bei den Befehlen, die eine Antwort des Messgeräts mit den Messdaten auslösen, um die Befehle „getOfflineDataNum“, der die Anzahl der abrufbaren Messdaten übermittelt, und „getOfflineData“, der diese Messdaten ausgibt. Diese beiden Befehle bestehen vollständig aus statischen Werten. Das feste Byte „B0“ wird gefolgt von einer Längenangabe des darauf folgenden Befehls, einem festen Byte „00“ sowie einem Nachrichten-Counter, der mit jeder Nachricht um *1* erhöht wird. Danach folgt der eigentliche Nachrichteninhalt sowie am Ende ein Byte einer Prüfsumme, die dadurch berechnet wird, dass alle Byte-Werte miteinander addiert werden. Die Prüfsumme wird in die Längenangabe nicht mit einbezogen. Bei beiden Befehlen beginnt der eigentliche Inhalt mit dem Gerätetyp, der in diesem Fall auf „A1“ festgelegt ist. Im Fall des Befehls zur Übermittlung der Anzahl neuer Messdaten folgt darauf ein fester Wert „40 01 00 00“, im Fall des zweiten Befehls ein fester Wert „46 01 00 00“ und anschließend jeweils das entsprechende Prüfsummen-Byte. Die genauen Befehle sind im Anhang in II.5 dargestellt.

Es ist zwar möglich Befehle oder einzelne Befehlsteile systematisch oder zufällig zu verändern, allerdings ist dies nicht notwendig. Es handelt sich bei den beiden Befehlen um

rein statische Befehle, sodass diese zum selbstständigen Einspielen so übernommen werden können. Beim Einspielen dieser Nachrichten antwortet das Messgerät jedoch nicht mit der Anzahl der Messdaten oder den Messdaten selbst, sondern gibt eine Fehlermeldung zurück, die im Quellcode der App als „ERR_COMM“ bezeichnet wird. Es ist also nicht ausreichend diese beiden Befehle abzusenden, um an Messdaten zu gelangen. Daher wurden die weiteren aufgenommenen Befehle vor der Datenabfrage näher analysiert.

Der erste Befehl der Kommunikation wird in der App auch als „getIdps“ bezeichnet und ist ein statischer Befehl. Er kann an das Gerät gesendet werden und dieses antwortet - wie erwartet - mit Informationen zu sich selbst. Danach folgen allerdings im Log zwei Befehle der App an das Gerät, die nicht vollständig statisch sind. Der erste Befehl ist der Befehl „Identify“, der nach dem zuvor erläuterten grundlegenden Nachrichtenaufbau eine 16 Byte große Zufallszahl enthält. Das Messgerät antwortet daraufhin mit einer Nachricht, die drei weitere jeweils 16 Byte große Werte enthält. Der erste Teil ist immer gleich (statisch), der zweite und dritte sind jeweils immer unterschiedlich.

Aus dem ersten statischen Teil $Wert_1$ wird ein Schlüssel S_1 generiert, indem $Wert_1$ mit einem weiteren Schlüssel S_2 verschlüsselt wird. S_2 wiederum besteht ebenfalls aus einem verschlüsselten Wert $Wert_{Ext}$, nämlich einem Wert aus einer extern eingebundenen Shared Library „libiHealth.so“, der auf dem mobilen Endgerät angefragt wird, verschlüsselt mit einem fest im Code einprogrammierten Schlüssel S_{Fest}. Mit S_1 wird dann der dritte Teil des empfangenen Befehls $Wert_3$ verschlüsselt und als Befehl nach dem bekannten Schema an das Messgerät gesendet. Der zweite Teil $Wert_2$ wird im Befehl nicht verwendet.

Als Verfahren für die Verschlüsselung wird jeweils XXTEA mit einem 128 Bit großen Schlüssel verwendet. Die Formel lautet also (mit $XXTEA_{Schlüssel}(Nutzdaten)$ als Algorithmus zum Verschlüsseln) wie folgt:

$$Verschlüsselter\ Wert = XXTEA_{S_1}(Wert_3) \tag{1}$$

$$S_1 = XXTEA_{S_2}(Wert_1) \tag{2}$$

$$S_2 = XXTEA_{S_{Fest}}(Wert_{Ext}) \tag{3}$$

Die einzige Unbekannte zum Generieren des gesamten Befehls ist also der Wert $Wert_{Ext}$, der aus dem externen Aufruf der Library resultiert. Denn die beiden Werte $Wert_1$ und $Wert_3$ sind in der empfangenen Nachricht des Messgeräts enthalten, S_{Fest} befindet sich im Quellcode der App und alle weiteren Schlüssel können mit Kenntnis von $Wert_{Ext}$ berechnet werden.

Die externe Library befindet sich ebenfalls unter dem Code der dekompilierten Android-App, ist jedoch eine kompilierte Datei eines C++-Codes. Beim Einbinden dieser Datei sowohl in ein Python-Script, als auch ein C-Script und ein C++-Script kommt es jedoch zu dem Problem, dass angeblich der Header der Datei fehlerhaft ist. Da die Datei aber offensichtlich vom Android-Gerät angenommen wurde - sonst hätte die iHealth-App ebenfalls einen Fehler ausgegeben -, wird eine eigene Android App geschrieben, die lediglich diese Library einbindet, die entsprechende Funktion „getKey“ aufruft und anzeigt. Bei dessen Funktionsaufruf stellt sich heraus, dass es sich um einen 16 Byte großen Schlüssel handelt, der auch bei mehrfachem Aufruf gleich bleibt. Es handelt sich also um einen statischen Schlüssel.

Somit sind alle Voraussetzungen gegeben, um den vom Blutdruckmessgerät gesendeten Befehl mit seinen drei 16 Byte Werten so zu verarbeiten, dass der daraus resultierende gesendete Befehl korrekt ist und vom Gerät angenommen wird. Danach ist es möglich die Befehle zum Abfragen der Anzahl an neuen Messdaten sowie zum Abfragen der Messwerte zu senden.

Es wurde ein Python-Script geschrieben, das alle Befehle hintereinander an das Blutdruckmessgerät sendet. Nach Senden des gültig verschlüsselten Befehls bleibt das Gerät weiterhin

an und akzeptiert diesen. Es beantwortet auch die darauf folgenden Befehle und somit ist es möglich die Messdaten abzufragen. Diese liegen dann in dem im Anhang in II.5 beschriebenen Schema vor.

Es ist also möglich als Angreifer an die Daten zu kommen, sofern der notwendige Schlüssel bekannt ist. Durch das feste Einprogrammieren des Schlüssels in den Code der App kann man jedoch davon ausgehen, dass Angreifer in der Lage sind diesen auszulesen. Ein benutzerspezifischer Schlüssel fließt nicht in die Berechnung mit ein. Die Vorgehensweise beim Abfragen neuer Messdaten ist in Abbildung 31 dargestellt.

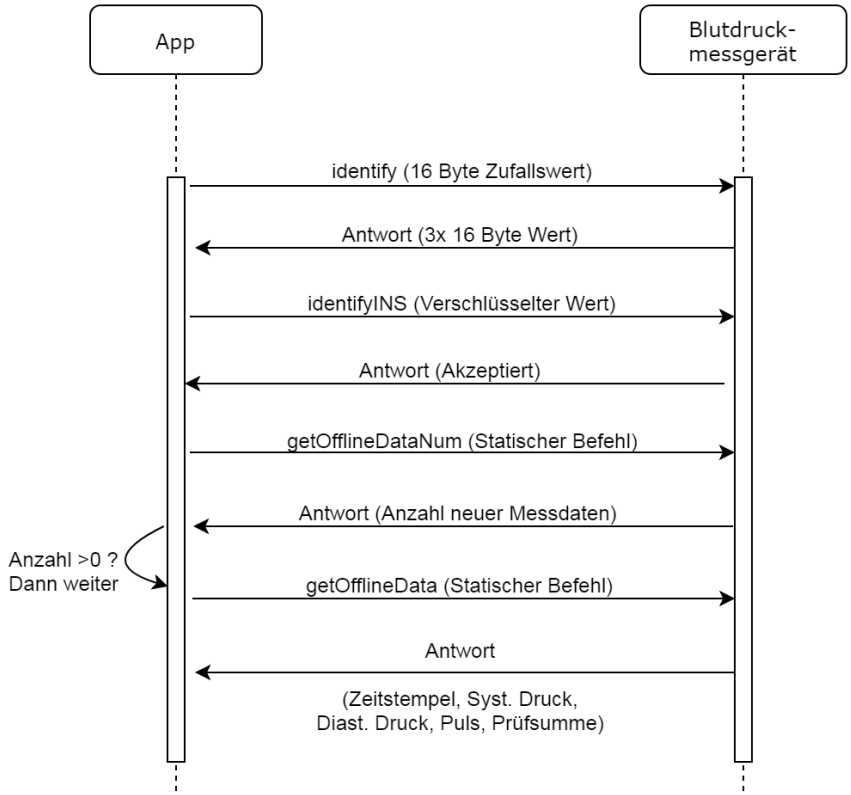

Abbildung 31: Vorgehensweise bei Datenabfrage neuer Messdaten

Die Prüfsumme am Ende eines jeden Befehls besteht lediglich aus einem Byte und ist die Summe aller im Befehl verwendeten Bytes. Somit ist sie und damit auch die gesamte Nachricht unbemerkt manipulierbar. Die Prüfsumme wird vom Gerät überprüft und bei fehlerhafter Prüfsumme oder falschem Befehl wird der Befehl nicht beantwortet und somit ignoriert.

Da das Blutdruckmessgerät mit einem Akku betrieben wird, ist es prinzipiell auch anfällig für Angriffe auf die Stromversorgung und könnte durch dauerhafte (bösartige) Kommunikation dazu gebracht werden, dass das Gerät ausgeht. Allerdings gibt es keine großen Konsequenzen dadurch. Das Gerät ist ohnehin zum einen nur in Betrieb, nachdem es vom Benutzer angeschaltet wurde und geht nach einer bestimmten Zeit wieder automatisch aus. Zum anderen ist es mit Bluetooth nur verbindbar, sofern Bluetooth ebenfalls über die Taste aktiviert wurde. Nach fünf Minuten geht auch diese Funktionalität wieder aus. Es gibt keine Variablen (z.B. Counter, Nonces), die beim Neustart zurückgesetzt würden und eine Sicherheitslücke bilden könnten. Der Counter der in den Nachrichten verwendet wird, wird standardmäßig ohnehin für jede neue Verbindung zurückgesetzt. Zudem ist er durch die fehlende Verschlüsselung und unzureichende Integritätsprüfung manipulierbar.

Die einzige Konsequenz durch einen Ausfall der Stromversorgung wäre, dass der Benutzer sein Blutdruckmessgerät nicht mehr einschalten kann, weder um Messungen durchzuführen, noch um Daten zu synchronisieren. Alle zuvor aufgenommenen Daten befinden sich in der App und erfordern zum Einsehen keine Verbindung zum Gerät.

Der Benutzer der App muss zwar vor der ersten Verwendung ein Benutzerkonto anlegen, dieses wird jedoch nur verwendet, um sich an der iHealth-Cloud anzumelden, in der die Messdaten gespeichert werden. Am Blutdruckmessgerät selbst ist keine Authentifizierung notwendig. Für einen bestimmten Befehl, der erforderlich ist, bevor Messdaten unverschlüsselt angefragt werden können, ist es jedoch erforderlich einen Schlüssel zu kennen, mit dem ein Wert aus einer Nachricht des Messgeräts verschlüsselt wird.

Durch einen verschlüsselten Befehl der App an das Gerät soll gewährleistet sein, dass nicht jeder beliebige Absender die Messdaten abfragen kann. Jedoch wird kein Schlüssel ausgetauscht, der bestimmte Absender authentisiert, indem diese ein bestimmtes Wissen über den Schlüssel haben. Vielmehr wird ein statischer Schlüssel - der nicht benutzerspezifisch, sondern immer gleich ist (zumindest bei Geräten des gleichen Typs) - verwendet, sodass der Befehl auch verschlüsselt werden kann, wenn ein Angreifer diesen Schlüssel kennt. Dieser kann mit etwas Aufwand aus der App bzw. einer darin enthaltenen Library ausgelesen werden. Eine wirkliche Authentizität gibt es also nicht.

Alle Benutzer mit Besitz der App können sich mit dem Messgerät verbinden und Messdaten abfragen. Es ist auch kein Besitz eines Schlüssels oder Wissen über ein Passwort notwendig, um mit dem Gerät zu kommunizieren, da dieser in der App fest einprogrammiert ist. Wird die App nicht verwendet, ist Wissen über diesen Schlüssel notwendig, um Befehle zu manipulieren. Zudem gibt es keine Möglichkeit das Blutdruckmessgerät in den Werkszustand zurückzusetzen.

In der App konnte keine Möglichkeit gefunden werden Firmware-Updates einzuspielen. Auch in der Bedienungsanleitung des Herstellers werden Firmware-Updates nicht erwähnt, sodass davon auszugehen ist, dass diese im Fall des iHealth View nicht möglich sind. Die aktuell auf dem Gerät installierte Firmware-Version ist 1.0.0, wie es bei der ersten Bluetooth-Antwort des Messgeräts übertragen und in der App angezeigt wird.

Die gesendeten Befehle der App zum Erhalt der Messdaten sind zu einem großen Teil statisch und es wird keine Verschlüsselung eingesetzt. Alle Befehle haben jedoch einen Counter zum Bestandteil. Dieser wird mit jeder Nachricht hochgezählt. Der Counter wird jedoch bei jeder neuen Verbindung wieder zurückgesetzt. Sofern der Counter der aufgenommenen Nachricht höher ist als der zuletzt gesendete Counter in einer legitimen Nachricht, wäre es theoretisch denkbar, zuvor aufgenommene Pakete erneut einzuspielen. Anderenfalls ist es jedoch auch möglich, den Counter zu manipulieren, sodass dieser höher ist als der zuletzt gesendete. Da der Counter bei jeder neuen Verbindung zurückgesetzt wird, wäre ein Einspielen in der Regel also immer möglich, da das Gerät von sich aus keine Befehle generiert, sondern nur auf Befehle antwortet. Somit bleibt der Counter also zunächst beim Wert *0*. Bei Einspielen einer Nachricht mit fehlerhaftem Counter wird keine Antwort gesendet.

Allerdings ist das Einspielen alter Nachrichten praktisch nicht erfolgreich, da die weiteren Befehle zum „Identifizieren“ notwendig sind, um eine Datenabfrage auslösen zu können. Anderenfalls wird diese mit einem Fehler beantwortet. In der vom Gerät gesendeten Nachricht mit den drei 16 Byte großen Werten ändert sich jedoch mit jeder Verbindung der letzte der drei Werte. Der daraus resultierende Befehl der App ändert sich ebenfalls, da dieser diese Antwort verarbeitet. Somit ist es nicht möglich einen veralteten Identify-Befehl einzuspielen. Das Blutdruckmessgerät beendet bei einem solchen Versuch die Verbindung, zeigt auf dem Display die Fehlermeldung „Er C“ an und geht aus.

Da es keine Authentifizierung gibt und prinzipiell alle Geräte mit Wissen über die Befehle und den festen Schlüssel in der Lage sind diese abzusenden, ist es nicht notwendig Geräte zu

klonen oder zu fälschen.

6.6.2 Ergebnis

Die Ergebnisse der Analyse werden in der folgenden Tabelle 6 dargestellt und zusammengefasst.

Tabelle 6: Zusammenfassung Analyse iHealth Blutdruckmessgerät

Funkprotokoll	Bluetooth (Klassisch)
Verbindungsaufbau	5 Minuten nach Tastendruck möglich
Stromversorgung	Akku + Ladekabel
Verschlüsselung	XXTEA-128 (nur einzelner Befehl)
Schlüsselaustausch	Fest einprogrammierter Schlüssel
Nachrichtenlänge/-aussehen	Rückschlüsse auf Inhalt möglich
Befehle wiedereinspielbar	Nein
Befehle veränderbar	Nein, aber vollständig nachstellbar
Fehlerhafte Nachrichten	Werden ignoriert, bei fehlerhaftem „Identify-Befehl“ Fehlermeldung und Ausschalten des Geräts
Integritätsprüfungen	Eigene (manipulierbar), CRC
Verhalten nach Neustart	Kommunikation wie üblich möglich
Authentifikation	Nein
Firmware Up-/Downgrades	Keine Updates möglich

Das Blutdruckmessgerät verwendet im Gegensatz zu den anderen untersuchten Geräten kein Bluetooth Low Energy, sondern klassisches Bluetooth. Die Kommunikation kann größtenteils (mit Ausnahme einer Nachricht) mit entsprechender Hard- und Software für Bluetooth mitgelesen werden, da keine Verschlüsselung eingesetzt wird. Somit sind auch die Messdaten des Benutzers (Blutdruck und Puls) mitlesbar. Um das Gerät aktiv zu manipulieren und Daten abzufragen ist es jedoch erforderlich ebenfalls eine verschlüsselte Nachricht zu senden, bei Fehlern in dieser geht das Gerät aus. Als Schlüssel wird jedoch ein statischer fest einprogrammierter Schlüssel verwendet, der von Angreifern ausgelesen werden kann. Somit war es insgesamt möglich, die Befehle der App nachzustellen und Messdaten zu erhalten. Es existiert zwar eine Prüfsumme, diese ist jedoch nicht gegen Manipulationen geschützt.

Das Gerät ist jedoch insgesamt nur in der Lage per Bluetooth zu kommunizieren, wenn zuvor eine Taste am Gerät länger gedrückt wurde, sodass ein Angreifer lediglich einen dreiminütigen Zeitraum hätte, um an Daten zu gelangen. Insgesamt lässt sich jedoch sagen, dass das Gerät durch die fehlende Verschlüsselung im Allgemeinen als unsicher bewerten ist. Mit Kenntnis über die fest einprogrammierten Schlüssel ist jeder Angreifer in der Lage das Gerät zu steuern und Messdaten zu erhalten.

6.7 ABUS Fahrradschloss

Ein weiteres Gerät, das per Bluetooth Low Energy gesteuert wird, ist das ABUS Fahrradschloss. Dieses befindet sich noch in der Entwicklungsphase, die Analyse wird mit einem Prototyp des Schlosses durchgeführt.

Das Schloss besteht aus einem Hauptteil (Schlosskörper), in dem unter anderem auch die Bluetooth-Komponenten verbaut sind und einem Metallbügel, der mit dem Hauptteil zusammengesteckt wird, um ein Fahrrad vor Diebstahl zu schützen. Außerdem ist ein akustischer Alarm verbaut, der bei größeren Bewegungen auslöst. Zunächst gibt es dann einen Voralarm, der durch vereinzeltes Piepsen auf sich aufmerksam macht und bei anhaltenden Bewegungen ertönt ein deutlicherer akustischer Alarm. Der Schlosskörper besitzt einen kleinen Schalter, um das Schloss in den Transportmodus zu versetzen, damit beim Transport des Schlosses nicht der Alarm durch Bewegungen ausgelöst wird.

Das Schloss kann über eine entsprechende App oder optional über eine kleine Fernbedienung gesteuert werden. Das Schloss und die zugehörige Fernbedienung sind in der Abbildung 32 dargestellt.

Abbildung 32: ABUS Fahrradschloss mit Fernbedienung

Das Schloss reagiert auf Bewegungen. Um das Schloss zu schließen, muss lediglich der Bügel eingesteckt werden. Nach wenigen Sekunden gibt es ein kurzes akustisches Signal und das Schloss verschließt sich, wenn es für eine kurze Zeit nicht mehr bewegt wurde. Um das Schloss zu öffnen ist ein Schütteln des Schlosses notwendig. Wenn sich der legitime Benutzer, der eine entsprechende Fernbedienung oder App auf einem mobilen Endgerät besitzt, in der Reichweite befindet und diese bedient, ertönt dann ein Piepton und das Schloss öffnet sich. In diesem Fall ist es auch möglich die Transportsicherung durch längeres Drücken des entsprechenden Knopfes am Schloss ein- oder auszuschalten. Auch danach ertönt ein kurzer Piepton. Befindet sich der legitime Benutzer nicht in der Nähe und das Schloss wird geschüttelt oder ist größeren Bewegungen ausgesetzt, gibt es einen Voralarm, der bei anhaltenden Bewegungen in einen aktiven Alarm übergeht.

Das Schloss wird mit einem eingebauten Akku betrieben, der mit einem Stecker an der Seite des Schlosses wieder aufgeladen werden kann.

Zum Zeitpunkt der Sicherheitsanalyse lag die App nicht vor. Daher konnte lediglich die Kommunikation zwischen der Fernbedienung und dem Fahrradschloss analysiert werden.

6.7.1 Sicherheitsanalyse

Zunächst werden auch bei dieser Analyse allgemeine Informationen über das Schloss gesammelt sowie auch über die Fernbedienung. Beide Geräte kommunizieren mit Bluetooth Low Energy. Die Fernbedienung soll in einer späteren Version in der App hinzugefügt und so mit dem Schloss gekoppelt werden können. In der vorliegenden Version ist dies noch nicht der Fall, die Fernbedienung ist jedoch bereits zuvor vom Hersteller mit dem Schloss verbunden worden und entsprechende Schlüssel für die Kommunikation sind bereits ausgetauscht, sodass beide Geräte miteinander kommunizieren können.

Das Schloss sendet durchgehend Advertising-Pakete im Abstand von einer Sekunde aus. Diese sind vom Typ „ADV_IND“, geben also an, dass eine Verbindung mit diesem Gerät möglich ist. Der Inhalt ist dabei immer gleich (der Name des Schlosses „ABLCM“ und eine Nummernfolge). Sobald die entsprechende Taste gedrückt wurde, sendet auch die Fernbedienung Advertising-Pakete aus, jedoch mit einem Abstand von etwa 20-30 Millisekunden. Auch diese Pakete sind vom Typ „ADV_IND“ und indizieren somit die Möglichkeit eines Verbindungsaufbaus. Der Inhalt dieser Pakete ist ebenfalls immer gleich und ähnlich dem des Schlosses („ABLHS“ und die gleiche Nummernfolge). Ein Scan mit „BLEAH“ wird vom Schloss nicht beantwortet und die Verbindung zu Beginn des Scans wieder beendet.

Beim Schütteln des Schlosses in verschlossenem Zustand und gleichzeitigem Aussenden der Advertising-Pakete der Fernbedienung, kommt eine Verbindung zwischen beiden Geräten zu Stande. Bei dieser werden zunächst verschiedene Attribute ausgelesen und im Anschluss ein Schreibbefehl gesendet, wodurch das Schloss sich öffnet. Danach wird die Verbindung wieder beendet.

Bei Initiieren der Verbindung gibt es keinen „Encryption Request“, der eine Verschlüsselung nach BLE-Standard einleitet. Es erfolgt die Kommunikation also (zumindest nach BLE-Standard) unverschlüsselt. Somit gibt es auch keinen Pairing-Prozess nach BLE-Standard, durch den zuvor entsprechende Schlüssel ausgetauscht wurden. Allerdings gibt es einen vom Hersteller mitgelieferten ausgedruckten QR-Code auf einer Karte, der einen verschlüsselten initialen Schlüsselaustausch ermöglicht. Ein temporärer Schlüssel ist demnach bereits im Werkszustand im Schloss vorhanden und kann in der App durch Einlesen des QR-Codes eingegeben werden.

Durch die Vorgehensweise des Einscannens eines QR-Codes kann der temporäre Schlüssel von Angreifern nicht durch Mitschneiden der initialen Kommunikation erlangt werden. Gesichert durch diesen temporären Schlüssel kann dann ein zufällig generierter Schlüssel ausgetauscht werden, der im folgenden für die Verschlüsselung der Kommunikation verwendet werden kann. In diesem Fall ist es also auch nicht durch einen Man-in-the-Middle-Angriff oder durch Erzwingen eines unsicheren Verfahrens (Downgrade) möglich, den temporären oder den ausgetauschten Schlüssel zu erhalten. Einen unverschlüsselten Schlüsselaustausch oder fest einprogrammierte Schlüssel gibt es ebenfalls nicht, da diese durch den mittels Schlüssel im QR-Code abgesicherten Transport nicht notwendig sind.

Beim Mitschneiden der Kommunikation beim Öffnen des Schlosses mit der Fernbedienung fällt auf, dass ein einziger Schreibbefehl auf das Handle *0x000B* abgesendet wird. Dieser besteht aus 16 Byte Daten, die dafür sorgen, dass das Schloss sich öffnet. Diese Daten variieren mit jedem Öffnen des Schlosses, sodass keiner der Befehle inhaltlich gleich aussieht. Die Länge des Befehls bleibt jedoch immer gleich. Rückschlüsse auf den Inhalt durch die Länge oder das Aussehen des Befehls sind also nur insofern möglich, als es nur diesen einen Schreibbefehl gibt, der das Schloss öffnet und keine weiteren Befehle dieses Typs. Es existieren keine weiteren Schreibbefehle. Inhaltlich sind jedoch keine weiteren Rückschlüsse möglich.

Welches Verfahren genau zur Verschlüsselung der Nachrichten eingesetzt wird, konnte ohne Dekompilieren der App nicht ermittelt werden. Der Aufbau des gesendeten Befehls

zum Öffnen des Schlosses kann aus der aufgenommenen Kommunikation nicht entnommen werden. Der gesamte Befehl ist verschlüsselt und ändert sich bei jedem Absenden, sodass es ohne Dekompilieren der App nicht möglich war die Bestandteile des Befehls ausfindig zu machen.

Um einen korrekten Befehl zum Öffnen nachzustellen, ist der zugehörige Schlüssel notwendig, der zuvor mit einem temporären Schlüssel verschlüsselt übertragen wurde. Es kann jedoch versucht werden eine entsprechende zuvor aufgenommene Nachricht systematisch oder zufällig zu verändern („Fuzzing“). Beim Einspielen fehlerhafter Nachrichten, die durch verändern zufälliger Bestandteile eines zuvor aufgenommenen Befehls erstellt wurden, werden diese durch das Schloss ignoriert. Das bedeutet, es gibt keine Reaktion des Schlosses und dieses bleibt geschlossen. Dies gilt sowohl für Befehle, die eine gültige Länge von 16 Byte hätten, sowie für Nachrichten mit beliebiger anderer Länge.

Es konnte nicht ermittelt werden, ob im verschlüsselten Befehl ebenfalls eine Integritätsprüfung enthalten ist. Auch dafür wäre es erforderlich das eingesetzte Verfahren zur Verschlüsselung und den Aufbau der verschlüsselten Nachrichten zu kennen.

Durch die Stromversorgung des Schlosses mit Akkus ist diese anfällig für Angriffe auf die Akkudauer. Nachdem diese leer werden, geht das Gerät aus. Jedoch gibt es dadurch keine weiteren Konsequenzen für den Benutzer, da das Gerät mechanisch weiterhin verschlossen bleibt. Nach Herstelleraussage öffnet es sich nicht, der Benutzer ist jedoch nicht in der Lage das Schloss mit Fernbedienung oder App in der Zeit zu bedienen, bis er es mit entsprechendem Ladekabel wieder aufgeladen hat. Je nach Verfahren zur Verschlüsselung wäre es denkbar, dass bei einem Neustart des Schlosses nach Wiederaufladen flüchtige Variablen wie Counter zurückgesetzt werden. Jedoch sind genauere Aussagen ohne den Quellcode zu kennen nicht möglich.

Authentizität ist dadurch gegeben, dass ein legitimer Benutzer ein Geheimnis besitzt, nämlich den Schlüssel mit dem die Kommunikation verschlüsselt wird. Diesen Schlüssel hat er dadurch erhalten, dass er im Besitz einer Karte mit einem temporären Schlüssel ist, mit der der Schlüssel gesichert übertragen werden konnte. Angreifer besitzen weder den temporären Schlüssel, noch den initial ausgetauschten Schlüssel und sind somit nur in der Lage entweder den Schlüssel oder die Befehle korrekt zu „raten“. Bei einer Schlüssellänge von 128 Bit (16 Byte) und einer Befehlslänge von 16 Byte ist dies in beiden Fällen jedoch mit einem sehr hohen Aufwand verbunden (wie im Beispiel des Eqiva Türschlossantriebs in 6.3.1 mehrere Jahre). Einen PIN oder ein Passwort muss der Benutzer zudem nicht eingeben.

Zum Zeitpunkt der Sicherheitsanalyse gab es noch keine Möglichkeit Firmware-Updates einzuspielen. Es ist jedoch geplant, dass diese Funktion im Laufe des Entwicklungsprozesses hinzukommt.

Dadurch dass sich bei jedem Öffnen der entsprechende Schreibbefehl immer unterscheidet, ist es nicht möglich Befehle aufzunehmen und später erneut einzuspielen. Beim Einspielen alter Nachrichten werden diese wie beim Einspielen fehlerhafter Nachrichten vom Schloss ignoriert.

Die Authentifizierung des legitimen Benutzers erfolgt durch den Besitz eines gültigen Schlüssels, der mit einem temporären bekannten Schlüssel sicher ausgetauscht wird. Es ist daher nicht notwendig Merkmale von Geräten zu klonen oder fälschen.

6.7.2 Ergebnis

In der folgenden Tabelle 7 werden die Ergebnisse der Analyse zusammengefasst.

Tabelle 7: Zusammenfassung Analyse ABUS Fahrradschloss

Funkprotokoll	Bluetooth Low Energy (BLE)
Verbindungsaufbau	Nach Schütteln des Schlosses
Stromversorgung	Akku + Ladekabel
Verschlüsselung	Ja, Verfahren unbekannt
Schlüsselaustausch	Verschlüsselte Übertragung (mit Schlüssel aus QR-Code auf Karte)
Nachrichtenlänge/-aussehen	Keine Rückschlüsse auf Inhalt möglich
Befehle wiedereinspielbar	Nein
Befehle veränderbar	Nein
Fehlerhafte Nachrichten	Werden ignoriert
Integritätsprüfungen	Unbekannt, CRC
Verhalten nach Neustart	Unbekannt
Authentifikation	Kenntnis über Schlüssel
Firmware Up-/Downgrades	Geplant, in Prototyp noch nicht vorhanden

Durch den Einsatz von Verschlüsselung ist es nicht möglich den genauen Befehlsinhalt mitzulesen. Da sich die Befehle bei jedem Absenden ändern, ist es ebenfalls nicht möglich gesendete Befehle aufzunehmen und später wieder einzuspielen. Die mit einem temporären Schlüssel auf einem QR-Code gesicherte Übertragung des Schlüssels für die Kommunikation ist vor dem Abhören der Kommunikation gesichert. Da Befehle entsprechend lang sind, ist ein Brute Force, um das Schloss zu öffnen oder den Schlüssel zu erhalten, mit einem sehr hohen Aufwand mehrerer Jahre verbunden.

Auch Einspielen fehlerhafter zufällig veränderter Nachrichten führte dazu, dass diese ignoriert wurden. Durch die fehlende Möglichkeit die App genauer zu analysieren, sind wenige Punkte offen geblieben. Insgesamt war es mit der durchgeführten Analyse jedoch nicht möglich das Schloss zu steuern, ohne Kenntnis über den verwendeten Schlüssel zu besitzen.

7 Evaluation

Mit der erarbeiteten Vorgehensweise zur Sicherheitsanalyse in der Heimautomatisierung war es möglich verschiedene Geräte zu analysieren. Diese wird im Folgenden evaluiert. Mit der erarbeiteten Sicherheitsanalyse war es bei allen Geräten möglich die Vorgehensweise sinnvoll anzuwenden und bei fünf Geräten war es möglich mit dieser Sicherheitsprobleme bei der Kommunikation per Funkprotokoll zu finden, mit denen es gelang die volle Kontrolle über die Geräte zu erhalten.

Es konnte zwar eine möglichst allgemeine Vorgehensweise entworfen werden, die unabhängig von bestimmten Funkprotokollen ist, jedoch muss diese unter Umständen zunächst noch angepasst werden, je nach zu untersuchenden Geräten und Funkprotokoll. Dabei könnte es nicht gleich ersichtlich sein, welche Punkte für ein bestimmtes Protokoll nicht relevant oder anwendbar sind. Zur Verbesserung könnte man also Punkte, die nicht bei jedem Protokoll anwendbar sind, anders kennzeichnen.

Zudem fiel während der Durchführung der Sicherheitsanalyse auf, dass es bei manchen Geräten sinnvoller sein kann die Reihenfolge der Vorgehensweise zu verändern, wie zum Beispiel die Punkte des Schutzziels Nicht-Vermehrbarkeit hinter dem der Integrität oder zusammen zu behandeln. Außerdem hängen die Schutzziele häufig miteinander zusammen, sodass eine strikte Trennung z.B. zwischen Vertraulichkeit und Nicht-Vermehrbarkeit nicht gemacht werden kann. Denn Verschlüsselungsalgorithmen, die auch Maßnahmen zur Nicht-Vermehrbarkeit, wie beispielsweise Counter, beinhalten, würden in diesem Fall zur Folge haben, dass die Nicht-Vermehrbarkeit ebenfalls bereits im Punkt Vertraulichkeit beschrieben wird.

Während der Anwendung der Sicherheitsanalyse konnte außerdem festgestellt werden, dass häufig ein Dekompilieren der Apps, die zur Steuerung der Geräte eingesetzt werden, weiterhelfen kann, wenn die Vorgehensweise der Hersteller beim Einsatz von Verschlüsselung oder teilweiser Verschlüsselung der Nachrichten nicht offensichtlich ist. Die (untersuchten) Apps sind für Android kompiliert und es gibt entsprechende Decompiler, die ein Dekompilieren des Codes in Java ermöglichen. Dadurch können Kenntnisse über die genauere Vorgehensweise der Hersteller gewonnen werden, da diese in der genauen Implementierung häufig nicht exakt an Standards gebunden sind. In den gängigen Funkprotokoll-Standards ist zwar eine Verschlüsselung vorgesehen, die Entscheidung diese einzusetzen liegt jedoch bei den Herstellern. Auf Grund der freien Verfügbarkeit der Android-Apps kann man davon ausgehen, dass Angreifer immer in der Lage sind eine solche App zu erhalten und zu dekompilieren. Ein Dekompilieren liefert allerdings nicht immer einen Erfolg, wenn es darum geht Geräte als Angreifer zu übernehmen. Denn wenn Schlüssel oder Variablen nicht fest von der App oder einem spezifischen Gerät abhängen, sondern von Eingaben eines Benutzers, Zufallswerten oder externen Informationsquellen (z.B. ausgedruckten QR-Codes), sind diese Informationen auch nicht über das Dekompilieren zu erhalten. Zumindest aber Kenntnisse darüber welche Verfahren zur Verschlüsselung eingesetzt werden oder den genauen Nachrichtenaufbau gibt ein Dekompilieren in der Regel schon.

Des Weiteren war es auch möglich ein Gerät, das klassisches Bluetooth statt Bluetooth Low Energy im Gegensatz zu den anderen Geräten verwendete, mit der gleichen Vorgehensweise zu untersuchen. Dabei wurde nicht die Kommunikation mit entsprechender Hard- und Software mitgeschnitten, da diese nicht für klassisches Bluetooth geeignet war, sondern Android-Logs analysiert, die die Kommunikation aufzeichneten. Auch hier war es möglich die gleiche Vorgehensweise zu verwenden wie bei Bluetooth Low Energy, obwohl das Medium hier ein anderes war.

Beim Punkt der Firmware-Updates in der Sicherheitsanalyse konnte nicht immer so vorge-

gangen werden wie es vergangene Angriffe beschrieben haben. Dies lag zum einen daran, dass Hersteller entweder gar keine Möglichkeit vorgesehen haben Firmware-Updates einzuspielen oder aber die Geräte für zukünftige Schulungszwecke auf ihrem aktuellen Firmware-Stand bleiben sollen, damit etwaige Sicherheitslücken mit Updates nicht geschlossen werden. Zum anderen war es nicht möglich zu überprüfen, ob Firmware-Downgrades einspielbar sind. Dies lag daran, dass die Hersteller - falls überhaupt - nur die aktuelle Firmware zur Verfügung stellen und alte Updates ggf. mit Sicherheitslücken nicht mehr verfügbar sind.

Potential für Erweiterungen bzw. Verbesserungen der Vorgehensweise bei der Sicherheitsanalyse gibt es dadurch, dass diese aktuell vor allem vergangene Angriffe auf die bekanntesten bzw. am weitesten verbreiteten Funkprotokolle ZigBee, Bluetooth und Z-Wave beinhaltet. Es gibt jedoch noch einige unbekanntere und/oder neuere Funkprotokolle, die mit in den allgemeinen Teil einfließen könnten. Allerdings gibt es über diese deutlich weniger Beschreibungen von vergangenen Angriffen oder Sicherheitslücken. Es könnten also noch weitere Angriffe eingebunden werden, die sich von den aktuell berücksichtigten unterscheiden und somit noch weitere Punkte in der Analyse ergänzt werden.

Außerdem könnte die Analyse um solche Angriffe erweitert werden, die nicht nur auf die Kommunikation per Funkprotokoll fokussieren. Schlüssel für die Kommunikation könnten Angreifer beispielsweise auch erhalten, indem Hardwarekomponenten ausgelesen werden. Solche Angriffe werden in der Sicherheitsanalyse aktuell nicht berücksichtigt, da der Fokus dieser Arbeit auf den Funkprotokollen liegt. Man könnte also weitere Punkte auch außerhalb des Funkprotokolls zusätzlich ergänzen.

Insgesamt war es jedoch mit dem aktuellen Stand der Analyse möglich die Geräte strukturiert zu analysieren und durch die Anwendung der immer gleichen Analyse gibt es eine gewisse Vergleichbarkeit zwischen den einzelnen Ergebnissen. Die Zusammenfassung der Analyseergebnisse anhand von einheitlichen Tabellen wird in der Vorgehensweise der Analyse zwar nicht beschrieben, bietet sich aber ebenfalls auf Grund der Vergleichbarkeit an.

Zusammenfassend lässt sich also sagen, dass die Vorgehensweise es möglich gemacht hat die untersuchten Geräte strukturiert und vergleichbar auf Sicherheitslücken oder -probleme zu analysieren und bietet das Potential durch weitere Angriffe, unter anderem auch auf seltenere oder andere als die bekanntesten Protokolle sowie Vorgehensweisen außerhalb der Kommunikation mit Funkprotokollen ergänzt zu werden.

8 Fazit

Verschiedene Angriffe unterschiedlicher Funkprotokolle aus der Vergangenheit in der Heimautomatisierung wurden vorgestellt und erklärt. Aus diesen ist dann eine verallgemeinerte Vorgehensweise erarbeitet worden, die eine strukturierte Sicherheitsanalyse von Geräten der Heimautomatisierung ermöglicht, die mit Funkprotokollen kommunizieren.

Es wurde zunächst eines der bekannten Funkprotokolle in der Heimautomatisierung - nämlich Bluetooth Low Energy - sowie Hard- und Softwarekomponenten ausgewählt, um eine solche Kommunikation aufzunehmen und analysieren zu können. Außerdem gab es eine Auswahl verschiedener Geräte unterschiedlicher Hersteller mit dem Ziel ein möglichst breites Spektrum der Heimautomatisierung abzudecken. Diese verwendeten alle Bluetooth Low Energy, wodurch eine gewisse Vergleichbarkeit der Ergebnisse und Wiederverwendbarkeit der Hard- und Softwarekomponenten zu Stande kam. Eines der Geräte verwendete - wie sich während der Analyse herausstellte - jedoch das klassische Bluetooth für die Kommunikation mit der App.

Alle Geräte wurden mit der zuvor erarbeiteten Vorgehensweise analysiert. In den bekannten Funkstandards, auch in Bluetooth Low Energy und Bluetooth, sind Maßnahmen vorgesehen, die eingesetzt werden sollten, um die Geräte bzw. ganze Infrastrukturen vor Angriffen zu schützen. Der Einsatz dieser Sicherheitsfunktionen ist jedoch häufig den Herstellern überlassen. Von sieben analysierten Geräten hatte lediglich eines die Sicherheitsfunktionen so implementiert wie sie im BLE-Standard vorgesehen sind (Pairing-Prozess, Verschlüsselung, Integritätsprüfung bei eQ-3 Heizkörperthermostat). Zwei weitere Geräte haben Sicherheitsmechanismen implementiert, verwenden BLE jedoch nur als Transportebene für ihre Nachrichten (Eqiva Türschlossantrieb und ABUS Fahrradschloss), ein Pairing findet nicht nach BLE-Standard statt. Stattdessen wird jeweils mittels ausgedruckten QR-Codes realisiert, auf denen sich ein Schlüssel befindet, der initial zur Verschlüsselung eines zufällig generierten Schlüssels verwendet wird.

Von den sieben analysierten Geräten gab es nur zwei Geräte, die so gegen Angriffe auf das Funkprotokoll abgesichert waren, dass es mit der erarbeiteten Vorgehensweise nicht möglich war, die Geräte vollständig unter Kontrolle zu bringen und zu steuern, sofern der verwendete Schlüssel nicht bekannt ist (Eqiva Türschlossantrieb und ABUS Fahrradschloss). Alle anderen fünf Geräte konnten beliebig gesteuert werden wie es ein legitimer Benutzer auch kann. Dies war entweder auf Grund von Implementierungsfehlern möglich (Nicht-Prüfen und akzeptieren von unverschlüsselten Verbindungen), in den meisten Fällen jedoch wegen fehlender Verschlüsselung, unsicherem Schlüsselaustausch oder Verwendung von statischen Befehlen.

Insgesamt lässt sich zum einen zusammenfassen, dass es mit der erarbeiteten Vorgehensweise möglich war strukturiert vorzugehen und viele Sicherheitsprobleme bei der Kommunikation zu erkennen. Zum anderen lässt sich feststellen, dass es insbesondere bei den Implementierungen der Sicherheitsmechanismen noch Verbesserungsbedarf auf Seiten der Hersteller von Geräten im Bereich der Heimautomatisierung gibt. Entweder existieren kaum Mechanismen die eine Übernahme des Geräts durch Angreifer verhindern oder diese sind nur unzureichend umgesetzt.

I Literaturverzeichnis

Albahar M.: *Bluetooth Pairing Security - Threats And Countermeasures.* Publications of the University of Eastern Finland. Dissertations in Forestry and Natural Sciences, No. 289, 2017

Bedner M. und Ackermann T.: *Schutzziele der IT-Sicherheit.* Datenschutz und Datensicherheit - DuD (Volume 34, Nr. 5), S.323-328, 2010

Bertko C. und Weber T.: *Home, Smart Home.* Carl Hanser Verlag München, 2017

BSI: *Technische Richtlinie, Kryptographische Verfahren: Empfehlungen und Schlüssellängen (BSI TR-02102-1), Version 2018-02.* Bundesamt für Sicherheit in der Informationstechnik (BSI), 2018. – `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf` (Letzter Zugriff 25.09.2018)

Cao X., Shila D., Cheng Y., Yang Z., Zhou Y. und Chen J.: *Ghost-in-ZigBee: Energy Depletion Attack on ZigBee-Based Wireless Networks.* IEEE Internet of Things Journal, Volume 3, Nr. 5, S.816-829, 2016

Celebucki D., Lin M. und Graham S.: *A Security Evaluation of Popular Internet of Things Protocols for Manufacturers.* IEEE International Conference on Consumer Electronics (ICCE), S.1-6, 2018

Deogirikar J. und Vidhate A.: *Security Attacks in IoT: A Survey.* International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud), S.32-37, 2017

Fouladi B. und Ghanoun S.: *Security Evaluation of the Z-Wave Wireless Protocol.* TechRepublic, 2013

Gierow H.: *Sicherheitslücken im Zigbee-Protokoll demonstriert.* Golem Newsticker vom 26.11.2015, 2015. – `https://www.golem.de/news/smart-home-sicherheitsluecken-im-zigbee-protokoll-demonstriert-1511-117657.html` (Letzter Zugriff 13.05.2018)

Hassan S., Bibon S., Hossain M. und Atiquzzaman M.: *Security threats in Bluetooth technology.* Computers & Security Volume 74, S.308-322, 2017

Heise Online: *Diffie-Hellmann (DH).* Heise Glossar, 2018. – `https://www.heise.de/glossar/entry/Diffie-Hellmann-397937.html` (Letzter Zugriff 08.07.2018)

Heydon R.: *Bluetooth Low Energy - The Developer's Handbook.* Pearson Education, Inc., 2013

Jasek S.: *GATTacking Bluetooth Smart devices.* 2016. – `http://gattack.io/whitepaper.pdf` (Letzter Zugriff 15.07.2018

Krausse M. und Konrad R.: *Drahtlose ZigBee-Netzwerke - Ein Kompendium.* Springer Vieweg Verlag, 2014

Marksteiner S., Jimenez V. J. E., Vallant H. und Zeiner H.: *An Overview of Wireless IoT Protocol Security in the Smart Home Domain.* Internet of Things Business Models, Users, and Networks, S.1-8, 2017

Morgner P., Mattejat S. und Benenson Z.: *All Your Bulbs Are Belong to Us: Investigating the Current State of Security in Connected Lighting Systems.* Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec, 2017

Olawumi O., Haataja K., Asikainen M., Vidgren N. und Toivanen P.: *Three practical attacks against ZigBee security: Attack scenario definitions, practical experiments, countermeasures, and lessons learned.* 14th International Conference on Hybrid Intelligent Systems, S. 199-206, 2014

OWASP: *Internet of Things Project.* 2018. – https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project (Letzter Zugriff 15.10.2018)

Ray A., Raj V., Oriol M., Monot A. und Obermeier S.: *Bluetooth Low Energy Devices Security Testing Framework.* IEEE 11th International Conference on Software Testing, Verification and Validation, 2018

Ronen E., O'Flynn C., Shamir A. und Weingarten A.: *IoT Goes Nuclear: Creating a ZigBee Chain Reaction.* IEEE Symposium on Security and Privacy (SP), S.195-212, 2017

Rose A. und Ramsey B.: *Picking Bluetooth Low Energy Locks from a Quarter Mile Away.* Merculite Security, DEFCON Vortrag, 2016. – https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEFCON-24-Rose-Ramsey-Picking-Bluetooth-Low-Energy-Locks-UPDATED.pdf (Letzter Zugriff 11.06.2018)

Saxena U., Sodhi J. S. und Singh Y.: *Analysis of security attacks in a smart home networks.* 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, 2017

Scherschel F.: *Licht an, Licht aus: ZigBee-Wurm befät smarte Glühbirnen.* Heise Newsticker Artikel vom 08.11.2016, 2016. – https://www.heise.de/security/meldung/Licht-an-Licht-aus-ZigBee-Wurm-befaellt-smarte-Gluehbirnen-3459004.html (Letzter Zugriff 26.03.2018)

Schröder H.: *Smart Home Monitor 2017: So smart ist Deutschland.* Webseite Digitaler Mittelstand, 2017. – https://digitaler-mittelstand.de/technologie/ratgeber/smart-home-monitor-2017-so-smart-ist-deutschland-42789 (Letzter Zugriff 26.03.2018)

Sokolov D.: *Deepsec: ZigBee macht Smart Home zum offenen Haus.* Heise Newsticker vom 21.11.2015, 2015. – https://www.heise.de/security/meldung/Deepsec-ZigBee-macht-Smart-Home-zum-offenen-Haus-3010287.html (Letzter Zugriff 24.05.2018)

SomersetRecon: *Bluetooth Attacks on Commercial-Grade Electronic Locks.* Recon Somerset Inc., 2016. – https://static1.squarespace.com/static/543effd8e4b095fba39dfe59/t/584f572a2994ca1083f4aba3/1481594669596/BluetoothAttacksonHighSecurityElectronicLocksWhitepaper.pdf (Letzter Zugriff 06.06.2018)

Tapplock Corp.: *Stay Protected - Notiz vom 12.06.2018.* 2018. – https://tapplock.com/notice/20180612/ (Letzter Zugriff 29.07.2018)

Tierney A.: *Totally Pwning the Tapplock Smart Lock.* PenTestPartners Penetration testing and security services, 2018. – `https://www.pentestpartners.com/security-blog/totally-pwning-the-tapplock-smart-lock/` (Letzter Zugriff 27.06.2018)

Tierney A.: *Z-Shave. Exploiting Z-Wave downgrade attacks.* PenTestPartners Penetration testing and security services, 2018. – `https://www.pentestpartners.com/security-blog/z-shave-exploiting-z-wave-downgrade-attacks/` (Letzter Zugriff 29.06.2018)

Uher J., Mennecke R. und Farroha B.: *Denial of Sleep Attacks in Bluetooth Low Energy Wireless Sensor Networks.* IEEE Military Communications Conference (MILCOM), 2016

Unwala I., Taqvi Z. und Lu J.: *IoT Security : ZWave and Thread.* IEEE Green Technologies Conference (GreenTech), 2018

Uskov A., Byerly A. und Heinemann C.: *Advanced Encryption Standard Analysis with Multimedia Data on Intel AES-NI Architecture.* International Journal of Computer Science and Applications, Vol. 13, Nr. 2, S.89-105, 2016

Yang Q. und Huang L.: *Inside Radio: An Attack and Defense Guide.* Springer Singapore, 2018

Yassein M. B., Mardini W. und Khalil A.: *Smart Homes Automation using Z-wave Protocol.* International Conference on Engineering & MIS (ICEMIS), 2016

ZigBee Alliance: *ZigBee Specification.* 2012

ZigBee Alliance: *ZigBee Over-the-Air Upgrading Cluster, Revision 23, Version 1.1.* 2014

Zillner T.: *ZigBee Exploited - The good, the bad and the ugly.* Cognosec GmbH, 2015

II Anhang

Bei allen folgenden Befehlen handelt es sich um jeweils einen zusammenhängenden Befehl. Die Leerstellen zwischen den einzelnen Befehlsteilen dienen lediglich der Lesbarkeit.

II.1 Befehle für Magic Blue Bulb

Lampe an-/ausschalten:

CC	23 (bzw. 24)	33
Fester Wert	ON (bzw. OFF)	Fester Wert

Helligkeit und Farbe ändern:

56	123456	00F0AA
Fester Wert	RGB-Wert	Fester Wert

Licht nach Mikrofon-Input:

78	12345678	00F0EE
Fester Wert	RGB-Wert	Fester Wert

Sonderfunktionen:

BB	2D	10	44
Fester Wert	Funktion (zw. 25 und 38)	Geschwindigkeit	Fester Wert

II.2 Befehle für Tapplock One

First Pair:

55AA81010400	AE9566C7	F503
CMD_PAIRING_FIRST_TIME	Fester Wert	Prüfsumme

Boot:

55AA92010C00	76916D6A	6ACEBA56	9E44AA6A	BA07
CMD_BOOTING	Key1	Key2	SerialNo	Prüfsumme

Regular Pair:

55AAB4010800	76916D6A	9E44AA6A	9005
CMD_PAIRING_REGULAR	Key1	SerialNo	Prüfsumme

Unlock:

55AA81020000	8201
CMD_UNLOCK	Prüfsumme

Factory Reset:

55AAE3010000	E301
CMD_FACTORY_RESET	Prüfsumme

II.3 Befehle für Basetech Raumthermostat

Temperatur einstellen (jede Zeile als separater Befehl):

4152
Fester Wert

14	00202A020000	11111100222222221200001111
Länge	versch. Werte	Fester Wert
	(2A = Temp. × 2 = 21 Grad)	

14	1111002222222212000011111110022222222
Länge	Fester Wert

14	1200001111111100222222221200001111111
Länge	Fester Wert

14	00222222221200001111111100222222221200
Länge	Fester Wert

14	00111111110022222222212000011	YYMMDDWWHH
Länge	Fester Wert	Zeitstempel

mit YY=Jahr, MM=Monat, DD=Tag, WW=Wochentag, HH=Stunde

08	MM	281A220A0134
Länge	Zeitstempel	Weitere Werte (z.B. Eco-Modus)

mit MM=Minute

4152
Fester Wert

II.4 Befehle für eQ-3 Heizkörperthermostat

Setzen der gewünschten Temperatur

41	2B
Fester Wert	Temperatur
	(2B = Temp. × 2 = 21.5 Grad)

Boost-Funktion

45	01
Fester Wert	Boost an/aus (01 = an)

II.5 Befehle für iHealth Blutdruckmessgerät

getOfflineDataNum:

B0	07	00	0B	A1	40010000	ED
Fester Wert	Länge	Fester Fester	Counter	Geräte-Typ	Befehlsinhalt	Prüfsumme

Antwort auf getOfflineDataNum:

A0	08	00	0C	A1	4001010000	EF
Fester Wert	Länge	Fester Wert	Counter	Geräte-Typ	Befehlsinhalt (01 = 1 neu)	Prüfsumme

getOfflineData:

B0	07	00	0F	A1	46010000	F7
Fester Wert	Länge	Fester Fester	Counter	Geräte-Typ	Befehlsinhalt	Prüfsumme

Antwort auf getOfflineData:

A0	10	00	10	A1	460100
Fester Wert	Länge	Fester Wert	Counter	Geräte-Typ	Befehlsinhalt

YYMMDDHHmm	ssddpp	2201	3C
Zeitstempel	Messung	Winkel/Änderung Messgerät	Prüfsumme

mit YY=Jahr, MM=Monat, DD=Tag, HH=Stunde, mm=Minute, ss=Systolischer Druck, dd=Diastolischer Druck, pp=Puls