IT-Security

Automatisierte Erkennung von Angriffen auf ZigBee-Netzwerke

Development of a Network Intrusion Detection System

Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Ingenieur/in

eingereicht von

Ing. Bernhard Bruckner, BSc

is201812

im Rahmen des

Studienganges IT-Security an der Fachhochschule St. Pölten

Betreuung

Betreuer/in: Dipl.-Ing. Oliver Eigner, BSc

Mitwirkung:

St. Pölten, 29. Mai 2022

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

*

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Diplomarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich sonst keiner unerlaubten Hilfe bedient habe.
- ich dieses Diplomarbeitsthema bisher weder im Inland noch im Ausland einem Begutachter/einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.
- diese Arbeit mit der vom Begutachter/von der Begutachterin beurteilten Arbeit übereinstimmt.

Der Studierende/Absolvent räumt der FH St. Pölten das Recht ein, die Diplomarbeit für Lehre- und Forschungstätigkeiten zu verwenden und damit zu werben (z.B. bei der Projektevernissage, in Publikationen, auf der Homepage), wobei der Absolvent als Urheber zu nennen ist. Jegliche kommerzielle Verwertung/-Nutzung bedarf einer weiteren Vereinbarung zwischen dem Studierenden/Absolventen und der FH St. Pölten.

Ort, Datum *Unterschrift*

Kurzfassung

ZigBee ist ein sehr weit verbreitetes Protokoll zur drahtlosen Übertragung von geringen Datenmengen und wird häufig in der Industrie, Medizintechnik sowie dem Smart Home Bereich verwendet. In vergangen Arbeiten wurden des Öfteren Angriffe auf diese Netzwerke durchgeführt, wobei die Auswirkungen für die Betroffenen meist fatal waren. So könnten beispielsweise smarte Türen ohne deren Wissen geöffnet, oder binnen Minuten die gesamte Lichtsteuerung einer Smart City durch Angreifer*innen übernommen werden.

Nach dem Erkennen dieser Problematik wurde nach einer Lösung gesucht, welche die Angriffe auf ZigBee-Netzwerke erkennen und die Besitzer*innen im Bedarfsfall benachrichtigen kann. Auf Grund der Absenz einer entsprechenden Lösung wurde im Zuge dieser Diplomarbeit eine solche entwickelt. Dabei wurden im ersten Schritt die gängigsten Angriffe auf ZigBee-Netzwerke bestimmt, wobei vier Angriffstypen identifiziert wurden. Danach folgte die Ableitung bestimmter Muster aus diesen Angriffen. Diese wurden in weiterer Folge in einer Datenbank gespeichert, wodurch es möglich ist jedes neue Paket mit diesen zu vergleichen. So wurde im Zuge dieser Arbeit ein Network Intrusion Detection System (NIDS) entwickelt, welches alle aufgezeichneten Pakete gegen die Muster in der Datenbank prüft. Wenn ein Angriff erkannt wird, kann eine beliebige Mailadresse benachrichtigt werden. Des Weiteren können die Besitzer*innen pro Angriffstyp festlegen, ob eine Benachrichtigung erhalten werden möchte. Um das System zu testen wurden zum einen Angriffe auf eine ZigBee-Testumgebung durchgeführt und zum anderen zuvor aufgezeichnete Angriffe aus pcap-Files verwendet.

Bei der Entwicklung des Systems wurde besonderes Augenmerk auf die Sicherheit von diesem gelegt. Aus diesem Grund müssen keine individuellen ZigBee-Keys im NIDS hinterlegt werden, da dessen Funktionalität auch ohne diese gegeben ist. Ein weiterer wichtiger Aspekt war die einfache Integration in bestehende ZigBee-Netzwerke. Somit wurde ein Programm entwickelt, um die Verwendung von ZigBee sicherer zu machen.

Abstract

ZigBee is a very widespread protocol for wireless transmission of small amounts of data and is frequently used in industry, medical technology and the smart home sector. In past work, attacks on these networks have often been carried out, and the consequences for those affected have usually been fatal. For example, smart doors could be opened without their knowledge, or within minutes the entire lighting control of a smart city could be taken over by attackers.

After the recognition of this problem, a solution was sought that could detect the attacks on ZigBee networks and notify the owners if necessary. Due to the absence of a corresponding solution, one was developed in the course of this thesis. In the first step, the most common attacks on ZigBee networks were determined, whereby four attack types were identified. This was followed by the derivation of certain patterns from these attacks. These were then stored in a database, making it possible to compare each new packet with them. Thus, a Network Intrusion Detection System (NIDS) was developed in the course of this work, which checks all recorded packets against the patterns in the database. When an attack is detected, any mail address can be notified. Furthermore, owners can specify per attack type if they want to receive a notification. To test the system, firstly, attacks were carried out on a ZigBee test environment and secondly, previously recorded attacks from pcap files were used.

During the development of the system, special attention was paid to the security of it. For this reason, no individual ZigBee keys need to be stored in the NIDS, as its functionality is also given without them. Another important aspect was the easy integration into existing ZigBee networks. Thus, a program was developed to make the use of ZigBee more secure.

Inhaltsverzeichnis

1 Einführung

Im privaten Umfeld ist schon des Längeren ein Trend in Richtung drahtloser Vernetzung erkennbar. Alle Geräte sollen jederzeit miteinander kommunizieren können, ohne diese zuvor über ein Kabel verbinden zu müssen. Sei es mittels WLAN (Wireless Local Area Network) um mit dem Smartphone oder Notebook Zugriff zum hauseigenen Netzwerk und in weiterer Folge zum Internet zu bekommen. So ist es auch oft bei anderen smarten Geräten oder Lampen gewünscht diese ohne große Renovierungsarbeiten in die Infrastruktur integrieren zu können. Ein weiterer Anwendungsfall dieser Technologie ist, dass oft erst zu einem späteren Zeitpunkt entschlossen wird beispielsweise die Lichtquellen des Eigenheims bequem per App steuern zu wollen. Dieser, anfangs groß wirkende, Aufwand beschränkt sich dann aber lediglich auf den Tausch der Glühbirnen oder der Schalter, um den gewünschten Komfort schaffen zu können. Das weit verbreitete Funkprotokoll namens ZigBee kommt hierbei in vielen Gebieten zur Anwendung. Dazu zählen beispielsweise [1]:

- Steuerung von Lichtquellen jeglicher Art
- Steuerung von Geräten in Gebäuden
- Automatisierte Steuerung von Anlagen in der Industrie
- Patientendatenübertragung in der Medizintechnik

1.1 Begriffsdefinition

ZigBee wird oft für Sensoren und Aktoren im Bereich der Heimautomatisierung verwendet. Dazu gehören beispielsweise Licht- und Regensensoren, Temperatursensoren, Rauchmelder, Fenster oder Türschlösser. Ein Nachteil der drahtlosen Kommunikation ist, dass diese auch leichter abgehört und manipuliert werden kann, da die Funkwellen von allen Geräten im Umkreis empfangen werden können. Weiters können auch Störsignale leichter gesendet werden, da hierfür lediglich die Empfänger in Reichweite sein müssen. Ein großer Vorteil von ZigBee ist das Design, wodurch sich dieses bestens für Geräte mit geringem Stromverbrauch eignet. So ist es beispielsweise möglich, batteriebetriebene Geräte auch mehrere Jahre mit derselben AA-Batterie zu betreiben. [2]

1.2 Relevanz des Themas

Um die Wichtigkeit von diesem Thema zu beleuchten, dienen folgend einige Statistiken über die Verbreitung von ZigBee zur Veranschaulichung. In diesen wird lediglich auf Elemente der Beleuchtung eingegangen, wobei dies nur einen Bruchteil des Einsatzbereiches darstellt und ZigBee auch für sehr viele andere Anforderungen verwendet wird. Als Beispiele seien hier die gesamte Heimautomatisierung und Medizin genannt.

Wie man der Statistik in Abbildung 1.1 entnehmen kann, wird weltweit im Jahr 2022 ein Absatz von rund 28,7 Millionen smarten Außenleuchten prognostiziert.

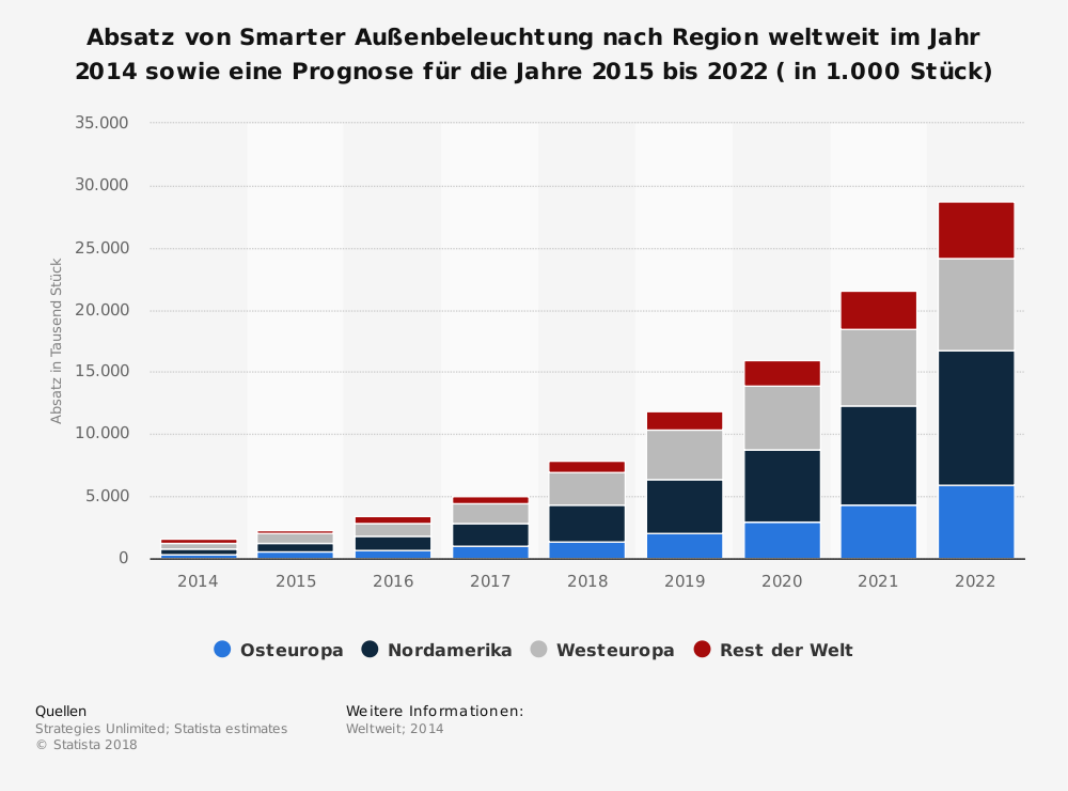

Abbildung 1.1: Globale Absatzentwicklung smarter Außenbeleuchtung bis 2022 [3]

Wie in Abbildung 1.2 ersichtlich werden rund 73%, dieser 28,7 Millionen Stück, LED Außenleuchten sein. Dies entspricht knapp 21 Millionen Stück, von denen laut Statistik in Abbildung 1.3 rund 41% das Funkprotokoll ZigBee verwenden werden.
Zusammengefasst werden somit im Jahr 2022 weltweit rund 8,6 Millionen smarte LED-Außenleuchten verkauft, welche den Funkstandard ZigBee verwenden.
Auf Grund der sehr großen Verbreitung mit steigender Tendenz wird ZigBee auch für Angreifer*innen

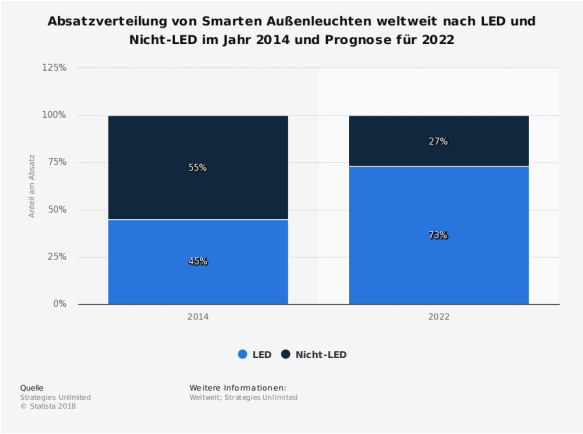

Abbildung 1.2: LED und Nicht-LED bis 2022 [4]

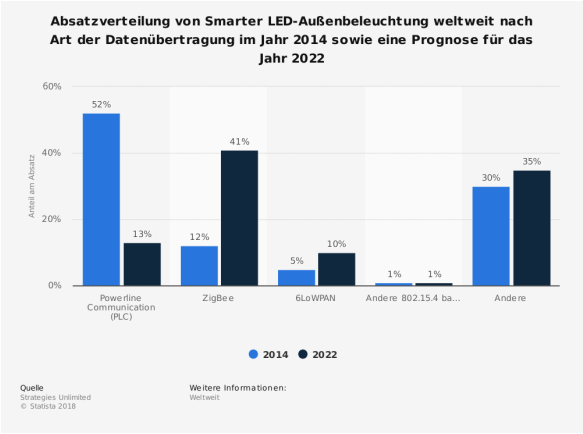

Abbildung 1.3: Datenübertragungssystem [5]

immer interessanter. Das Problem hierbei ist, dass je mehr ZigBee-Geräte verwendet werden desto größer ist der potenzielle Nutzen für diese sich damit zu beschäftigen. Vor allem in der Industrie und Medizintechnik sollte der Integration von ZigBee-Netzwerken in der Risikobewertung des Unternehmens besonderes Interesse geschenkt werden. In einer gut geschützten Infrastruktur würde sich dabei ein schlecht konfiguriertes ZigBee-Netzwerk als leichte Einfallstür für potenzielle Angreifer*innen anbieten. Da dieses auch meistens via LAN mit dem Netzwerk verbunden ist können sich Angreifer*innen im Worstcase in weiterer Folge in diesem ausbreiten und auf Firmendaten zugreifen. Ein weiterer Umstand ist, dass im Bereich der Heimautomatisierung ein Großteil der Benutzer*innen darauf vertraut, dass die angeschaffte Lösung basierend auf dem Stand der Technik als sicher gilt. Eine sicherheitsrelevante Bewertung ist diesen meist auf Grund von fehlendem Fachwissen nicht möglich, wodurch man sich auf die herstellerspezifischen Angaben verlassen muss.

1.3 Stand der Technik

Zum Zeitpunkt des Verfassens dieser Arbeit ist die aktuelle Version ZigBee3.0. Es gibt einige wissenschaftliche Arbeiten, welche Angriffe auf dieses Protokoll beschreiben und diese auch durchführen. Ich bin jedoch während meiner Recherche auf keine Arbeit gestoßen, welche sich darauf fokussiert die bekanntesten Angriffe zu erkennen und hinterlegte Kontaktdaten über das Auftreten dieser zu informieren. In Zeiten wo es bei der Überwachung von IP-Netzwerken schier grenzenlose Möglichkeiten mittels NIDS, NIPS, SIEM, WAF, Firewalling et cetera gibt, scheint punkto ZigBee noch Nachholbedarf zu bestehen.

G. Lee und sein Team stellen in ihrem Paper [6] einen Ansatz vor, um Sibyl-Angriffe in einem ZigBee-

Netzwerk zu erkennen und darauf zu reagieren. Bei diesem Angriff kompromittieren Angreifer*innen einen Sensor und versuchen in weiterer Folge mehrere virtuelle Identitäten im Netzwerk zu registrieren. Dadurch können diese mehrere gültige Nachrichten mit unterschiedlichen Source-Adressen generieren, ohne einen Denial-of-Service auszulösen. Beispielsweise könnten Angreifer*innen einen Sensor eines Feueralarmsystems übernehmen und mit diesem, neue virtuellen Identitäten im System erstellen. Diese werden in weiterer Folge beim Server als neue Sensoren registriert. Der Server des Alarmsystems hat hierbei keine Möglichkeit, diese als gefälscht zu erkennen. Die Angreifer*innen können anschließend von den hinzugefügten Identitäten mehrere Fehlalarme an den Server senden, welcher in weiterer Folge auf diese reagieren wird, obwohl gar kein Brand stattgefunden hat. Um die böswilligen Knoten zu identifizieren, messen Lee und sein Team die Zeit, welche die Knoten benötigen um auf eine Anfrage zu reagieren. In weiterer Folge blockieren sie sämtlichen Traffic von den, als böswillig identifizierten, Sensoren.

D. Akestoridis und P. Tague haben in ihrer Arbeit ein verteiltes Monitoring System zur Überwachung der Sicherheit von ZigBee Netzwerken entwickelt. Weiters haben sie mit einem Angriff dargestellt, wie eine 3 Volt CR2450 Batterie innerhalb von 16 Stunden vollständig aufgebraucht werden kann. Ihr entwickeltes System namens HiveGuard besteht aus einem Datenbankserver, einem Notification Server und Wireless Intrusion Detection Sensoren. Dabei wurden erstere in JavaScript und die Sensoren in Python entwickelt. Die einzelnen Komponenten kommunizieren untereinander über HTTP Requests. Sie konzentrieren sich auf das Erkennen folgender Angriffe: PAN ID Konflikt, Insecure Rejoin Requests, Key Leakages und niedrige Batteriestände von Geräten. [7]
Im Unterschied dazu werden vom im Zuge dieser Arbeit entwickelten NIDS noch zusätzlich Device Hijacking sowie Replay Angriffe erkannt. Weitere Unterschiede findet man bei der Herangehensweise an das Problem, den verwendeten Technologien sowie der Hardware.

F. Sadikin and S. Kumar entwickelten im Zuge ihrer Arbeit [8] ein IDS (Intrusion Detection System) welches zum einen regelbasiert und zum anderen mittels Machine Learning Algorithmus versucht bekannte sowie unbekannte Angriffe auf ZigBee basierte IoT Systeme zu erkennen. Dabei konzentrieren sie sich auf folgende Ziele von Angreifern und Angreiferinnen:

- Reconnaissance: Darunter versteht man das Auslesen von Informationen aus Geräten, welche für spätere Angriffe verwendet werden können.
- Denial-of-Service: Mittels DoS Angriff wird die Funktionalität der Services der Geräte eingeschränkt oder verhindert.

- Malicious Control: Hierbei versuchen die Angreifer*innen sich unautorisiert Zugriff zu einem Gerät zu verschaffen, um dieses missbräuchlich verwenden zu können.
- Device Hijacking: Beim Device Hijacking Angriff übernehmen die Angreifer*innen das ZigBee-Gerät, wobei die Benutzer*innen den Zugriff auf das Gerät verlieren.

Ein Unterschied zu meiner Arbeit liegt in der Erkennung der Angriffe. Hierbei verfolgen diese die Methode zur Erkennung eines Angriffs via RSSI (Received Signal Strength Indication). RSSI ist hierbei ein bestimmtes Muster, welches sich je nach Entfernung zwischen Sender und Empfänger unterscheidet. Anhand dieser Messwerte kann bestimmt werden, ob der soeben empfangene Befehl dasselbe RSSI Muster aufweist wie die zuvor als richtig angelernten Befehle oder ob dieser von einem anderen Gerät gesendet wurde.

Q. Sun und seine Kollegen versuchen in deren Arbeit Spoofing Angriffe mittels Machine Learning Algorithmen anhand von Informationen des Physical Layers zu erkennen. Diese verwenden dazu unter anderem RSS (received signal strength) Daten für das Trainieren des Modells. Die Erkennungsrate und Präzision von jeweils über 90% zeigen den Erfolg ihrer Arbeit. [9]

Sohel Rana und sein Team entwickelten in ihrer Arbeit ein Framework zur Verbesserung der Sicherheit in ZigBee Netzwerken für Monitoring in IoT Plattformen. Dieses Framework ist zwischen APL (Application Layer) und APS (Application Support Sublayer) angesiedelt. Im Zuge einer Fallstudie integrierten sie ihr Framework in das Sicherheitssystem eines Büros. Dabei zeigen die Ergebnisse, dass mehrere potenzielle Angriffe erkannt werden können. Dazu zählen physische Angriffe, Replay Angriffe und Flooding Angriffe. [1]
Der Unterschied zu meiner Arbeit liegt darin, dass hierbei die bestehende ZigBee-Architektur verändert wurde. Mein System hingegen funktioniert mit der bestehenden ZigBee Spezifikationen und kann aus diesem Grund einfach in ZigBee-Netzwerke integriert werden.

1.4 Forschungsfragen

Ist es generell möglich Muster in Angriffen auf ZigBee-Netzwerke zu erkennen?
Wie kann man Angriffe klassifizieren, automatisiert erkennen und die Besitzer*innen im Bedarfsfall benachrichtigen?

1.5 Methodik

Nachfolgend wird die Methodik und somit die Struktur der Arbeit näher beschrieben. Anfangs wird ein genereller Überblick über ZigBee sowie dessen Verbreitung gegeben. Nachdem auf die unterschiedlichen Topologien sowie den Protokoll Stack eingegangen wurde, werden die Applikationsprofile und zugehörigen Eigenheiten erläutert. Eine umfassende Betrachtung der sicherheitsrelevanten Aspekte bietet Kapitel 3. Dabei werden neben den ZigBee-Keys und deren Funktion in den jeweiligen Layern auch mögliche Risiken bei der Verwendung von ZigBee gezeigt. Im darauf folgenden Kapitel 4 werden die zuvor beleuchteten Sicherheitsrisiken in verschiedenen Angriffsszenarien verdeutlicht. Einen weiteren Teil dieses Kapitels bilden Angriffe theoretischer Natur. Den deutlichen Mehrwert dieser Arbeit beinhalten Kapitel 5 sowie Kapitel 6. Ersteres erweitert die zuvor vorgestellten Angriffe um die Verwendung des entwickelten Systems und gibt einen Überblick über dessen Entwicklung. Dabei werden die Vorteile für die Benutzer*innen sowie dessen Funktionsweise verdeutlicht. Zweiteres beschäftigt sich mit der Beschreibung der durchgeführten Tests und den daraus resultierenden Ergebnissen.

1.5.1 ZigBee Network Intrusion Detection System

Da während meiner Recherche keine Arbeit gefunden wurde, welche sich mit dem automatisierten Erkennen von den bekanntesten Angriffen auf ZigBee-Netzwerke beschäftigt wird im Zuge dieser Diplomarbeit ein solches ”ZigBee Network Intrusion Detection System” (NIDS) entwickelt. Dabei werden im ersten Schritt die aus der Literatur bekannten Angriffe analysiert und nach Möglichkeit in bestimmte Angriffstypen zusammengefasst. Folgende wurden dabei definiert und werden vom NIDS erkannt:

- Insecure Rejoin
- Übernahme von Geräten
- Replay
- Netzwerk-Key Sniffing

Das Ergebnis dieser Zusammenfassung dient in weiterer Folge als Grundlage für eine Analyse, in welcher bewertet wird ob diesen Angriffen eindeutige Muster zugeordnet werden können. Dabei wird versucht das Auftreten solcher Angriffe an bestimmte Vorgänge oder einzelne Befehle im ZigBee-Traffic zu koppeln. Im Betrieb des NIDS werden alle ZigBee-Pakete aufgezeichnet und im Zuge deren Verarbeitung und Speicherung gegen die Angriffsmuster geprüft. Ist die Prüfung erfolgreich, so wurde ein möglicher Angriff auf das System erkannt, wobei je nach Konfiguration eine Benachrichtigung versendet

werden kann. Das entwickelte ZigBee Network Intrusion Detection System ist einfach in bereits bestehende ZigBee-Netzwerke zu integrieren und kann an individuelle Gegebenheiten angepasst werden. Der große daraus resultierende Vorteil ist, dass ein Angriff nicht mehr unerkannt bleibt, sondern die Besitzer*innen des Systems sofort darüber informiert werden um notwendige Gegenmaßnahmen ergreifen zu können.

2 ZigBee Standard

2.1 Allgemein

Die ZigBee Alliance arbeitete eng mit der IEEE (Institute of Electrical and Electronics Engineers) an der Spezifikation des Protokollstapels. ZigBee baut auf dem Standard 802.15.4 auf, in welchem die Übertragungsverfahren für geringe Datenmengen definiert sind. In diesem, von der IEEE spezifizierten WPAN (Wireless Personal Area Network) Standard, werden die beiden untersten Schichten des OSI-Modells, nämlich der Physical-Layer und Data-Link-Layer, definiert. Kooperierend dazu entwickelt die ZigBee Alliance die darüberliegenden Schichten wodurch eine Interoperabilität zwischen verschiedenen Geräten von unterschiedlichen Herstellern gewährleistet wird. ZigBee ist ein Netzwerkprotokoll zur drahtlosen Übertragung geringer Datenmengen, wobei der Hauptfokus dieser Technologie weiters in der Herstellung einer kostengünstigen Funkverbindung und einem niedrigem Energieverbrauch liegt. Auf Grund der Protokolleigenschaften kann es auch zu Mesh-Netzwerken verbunden werden, wodurch es besonders für Automatisierung- und Remotesteuerung von Systemen jeglicher Art geeignet ist. Im Unterschied zu Bluetooth zielt ZigBee darauf ab geringere Datenmengen zu übertragen. Besonders häufig wird dieser Standard auch in Smart Home Umgebungen wie beispielsweise zur Steuerung von Licht verwendet. [10]
Seit dem Jahr 2012 ist es möglich, dass man Produkte von der ZigBee Alliance zertifizieren lässt. Somit ist sichergestellt, dass ein Produkt bestimmte Anforderungen erfüllt. Seit der aktuellen Version ZigBee 3.0 müssen Produkte mindestens die Anforderungen der Spezifikation ZigBee Pro 2015 erfüllen. ZigBee Pro 2017 beinhaltet zusätzlich die Unterstützung von zwei Frequenzbändern gleichzeitig. [11]

2.1.1 ZigBee Frequenzbereich

ZigBee verwendet das freie 2.4 GHz Frequenzband beziehungsweise je nach Region 915 MHz in Amerika oder 868 MHz in Europa. Je nach Frequenzband sind auch unterschiedliche Datenraten möglich, wobei im Falle von 2.4 GHz theoretisch 250kbit/s erreicht werden können. Im Gegensatz dazu sind bei 915 Mhz 40kbit/s möglich. Im 868 MHz Bereich werden die theoretischen Datenraten 20kb/s, 100kb/s und 250kb/s unterstützt. In den verschiedenen Frequenzbereichen wird zum Einen DSSS (Direct Se-

	ZigBee	Wi-Fi	Bluetooth
Distanz	10 - 100 Meter	50 - 100 Meter	10 - 100 Meter
Netzwerktopologie	Ad-hoc, Stern, Baum, Mesh	Point-to-hub	Ad-hoc, sehr kleine Netzwerke
Frequenzband	868 MHz, 2.4GHz	2.4 GHz und 5 GHz	2.4 GHz
Komplexität	gering	hoch	hoch
Stromverbrauch	sehr gering	hoch	mittel
Maximale Anzahl von Nodes	65.000	2.007	8

Tabelle 2.1: Unterschied zwischen ZigBee, WLAN und Bluetooth [11]

quence Spread Spectrum) als Frequenzspreizverfahren und zum Anderen OQPSK (Offset Quadrature Phase Shift Keying) als Modulationsverfahren verwendet. [2]

2.1.2 Unterschiede zu WLAN und Bluetooth

In Tabelle 2.1 werden die Unterschiede zu WLAN und Bluetooth aufgelistet. Dieser kann entnommen werden, dass sich ZigBee im Gegensatz zu den anderen beiden Funkprotokollen durch einen sehr geringen Stromverbrauch der Geräte und einer sehr großen Anzahl an möglichen Teilnehmern auszeichnet. Ein weiterer Unterschied ist, dass die verwendete Netzwerktopologie je nach Bedarf ausgewählt werden kann. Abschließend macht ZigBee die Einfachheit des Standards zu einem sehr beliebten und weitverbreiteten Protokoll im IoT Umfeld. [11]

2.2 ZigBee Protokoll Stack

Der ZigBee Protokoll Stack besteht aus vier Layern, wobei die beiden unteren Layer, nämlich der PHY- und MAC-Layer im IEEE 802.15.4-2003 Standard definiert sind. Die beiden darüber angesiedelten Layer hingegen, also NWK- sowie APL-Layer, werden durch den ZigBee Protokoll Stack beschrieben. Im IEEE 802.15.4 sind Funktionen für die Sicherstellung von Vertraulichkeit und Authentizität der Daten sowie dem Schutz vor Replay Angriffen umgesetzt. Darauf basierend wird die Funktionalität durch die beiden ZigBee-Layer um jene des Schlüsseltausches sowie der Verschlüsselung erweitert. [12] In Abbildung 2.1 wird diese Verteilung der Zuständigkeiten dargestellt.

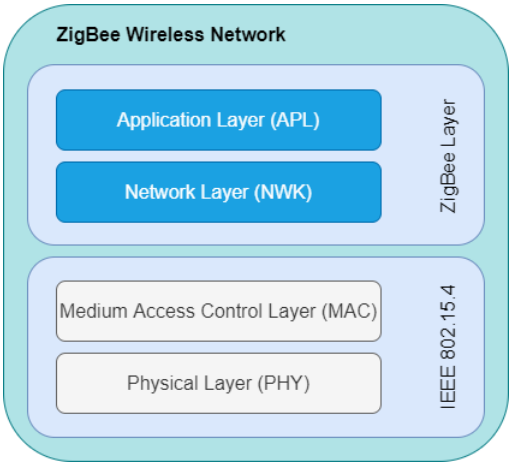

Abbildung 2.1: ZigBee Protokoll Stack [13]

Im Folgenden wird auf die einzelnen Layer im Detail eingegangen und deren Funktion erläutert.

2.2.1 Physical-Layer (PHY)

Der Physical-Layer beschreibt als unterster Layer des Modells die Schnittstelle zwischen Hardware und MAC-Layer und ermöglicht so zweiterem den Zugriff auf das Übertragungsmedium (Funk). In diesem sind die Funktionen für das Senden und Empfangen der Hochfrequenz Signale angesiedelt. Dazu gehört neben der Modulation und Demodulation der Signale auch das Filtern von Noise. Weiters gehört das Aktivieren und Deaktivieren der Transceiver, die Ermittlung der Verbindungsqualität, die Auswahl der Channels sowie das Versenden und Empfangen von PPDU Paketen (PHY protocol Data Unit) über das physische Medium zu den Funktionen des PHY-Layers. [14] Diese PPDU Pakete können in drei Teilbereiche unterteilt werden, wobei deren Struktur Abbildung 2.2 entnommen werden kann. [10]:

- SHR: dient zur Synchronisierung des Empfangsgeräts
- PHR: enthält Informationen über die Länge des Frames
- PHY Payload: enthält den MAC sublayer Frame

Octets: 4	1	1		variable
Preamble	SFD	Frame Length (7 bits)	Reserved (1 bit)	PSDU
SHR		PHR		PHY payload

Abbildung 2.2: ZigBee PPDU Format [10]

Ein Unterscheidungsmerkmal der verschiedenen Frequenzbereiche sind die zur Verfügung stehenden ZigBee-Channel. Im 868 MHz Bereich kann hierbei nur ein Channel verwendet werden, wohingegen bei Verwendung des 2.4 GHz Bereichs 16 Kanäle für die Kommunikation zur Verfügung stehen. Im 915 MHz Frequenzbereich kann wiederum auf zehn Channel zurückgegriffen werden [10].

2.2.2 Medium Access Control Layer (MAC)

Der MAC-Layer ist für eine zuverlässige Kommunikation zwischen zwei benachbarten Teilnehmern zuständig und beinhaltet die Adressierung von MPDU Paketen (MAC protocol data unit). Dadurch ist es möglich Pakete an eine bestimmte Zieladresse zu senden und beim Empfang festzustellen von welchem Teilnehmer diese kommen. Wie in Unterabschnitt 2.2.1 beschrieben werden die MPDU Pakete als Payload in die PPDU Pakete gekapselt. Ebenso kümmert sich dieser Layer beispielsweise um den Zugriff auf den Channel, die Vermeidung von Kollisionen, die Validierung der Frames, die Bestätigung des Erhalts der Frames et cetera. [15] Ein MPDU Paket besteht aus folgenden Komponenten, wobei die jeweilige Länge Abbildung 2.3 entnommen werden kann [10]:

- MHR bestehend aus Frame Control, Sequence Number und den Adressinformationen
- MAC Payload
- MFR (Frame Checksum)

<table>
<tr><td>Octets: 2</td><td>1</td><td>0/2</td><td>0/2/8</td><td>0/2</td><td>0/2/8</td><td>variable</td><td>2</td></tr>
<tr><td rowspan="2">Frame control</td><td rowspan="2">Sequence Number</td><td>Destination PAN identifier</td><td>Destination address</td><td>Source PAN identifier</td><td>Source address</td><td rowspan="2">Frame payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Adressing fields</td></tr>
<tr><td colspan="6">MHR</td><td>MAC payload</td><td>MFR</td></tr>
</table>

Abbildung 2.3: ZigBee MPDU Format [10]

Auf diesem Layer wird CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) verwendet, um beim Zugriff auf das Trägermedium Kollisionen möglichst gut zu vermeiden. Bei diesem Algorithmus wird vor der Datenübertragung das Trägermedium abgehört. Dadurch kann sichergestellt werden, dass zum Zeitpunkt des Sendens der Daten kein anderes Gerät am entsprechenden Kanal sendet. [10] Die sicherheitsrelevanten Informationen auf MAC-Layer Ebene sind in einem zusätzlichen Subheader, dem ”Auxiliary Security Header”, definiert. Hierzu zählen beispielsweise die verwendete Verschlüsselung und Informationen über die Art wie Frames geschützt werden. Dieser Header wird nur dann in den

Frame inkludiert, wenn das zugehörige ”Security Enabled Flag” aktiv ist. Weiters gibt es Attribute, die man in diesem Layer setzen kann, um eine möglichst flexible Sicherheitskonfiguration zu verwenden. Der MAC-Layer kümmert sich nicht um die Authentifizierung oder den Schlüsseltausch sondern vertraut darauf, dass dies von den darüberliegenden Layern erledigt wird. [12, 16]

2.2.3 Network-Layer (NWK)

In diesem Layer sind die für das Netzwerk zuständigen Services angesiedelt. So kümmert sich der Netzwerk-Layer beispielsweise um das Routing sowie das Management des gesamten ZigBee-Netzwerks, indem er die vom MAC-Layer zur Verfügung gestellten Funktionen aufruft. Dafür werden eigene ZigBee-Adressen verwendet, welche sich von den MAC Adressen unterscheiden. Des Weiteren ermöglicht der NWK-Layer das Entdecken von ZigBee-Netzwerken und den Beitritt zu diesen. Durch dessen Funktionen ist es außerdem möglich ein ZigBee-Netzwerk in Form eines Mesh Netzwerks zu betreiben. [12, 15] In Abbildung 2.4 wird das Format von NPDU Frames gezeigt [16]:

- NWK Header bestehend aus Frame Control, Radius, Sequence Number, Multicast Control, Source Route Subframe und den Adressinformationen.

  Das Feld ”Frame Control” beinhaltet Informationen wie die Version des Protokolls, die Information ob es sich um einen Multicast Frame handelt und ein Security Flag, welches angibt ob die Sicherheit auf NWK-Layer aktiviert ist.

  ”Radius” gibt hierbei die Anzahl an Hops an, welche das Paket weiterleiten dürfen und begrenzt somit die Reichweite.

  ”Source Route Subframe” beinhaltet eine Liste von Relays, welche zur Weiterleitung der Frames verwendet werden. Diese Liste ist absteigend nach der Nähe zum Zielgerät sortiert.

- Payload

Octets: 2	2	2	1	1	0/8	0/8	0/1	Variable	Variable
Frame control	Destination address	Source address	Radius	Sequence Number	Destination IEEE address	Source IEEE address	Multicast control	Source route subframe	Frame payload
NWK Header									Payload

Abbildung 2.4: ZigBee NPDU Format [17]

2.2.4 Application-Layer (APL)

Im APL werden die Formate der Datenpakete spezifiziert. Des Weiteren wird auf diesem Layer den Applikationen ein Service zur Verfügung gestellt um auf die Daten zugreifen zu können. [12] Wie in

Abbildung 2.5 ersichtlich besteht ein APL Frame prinzipiell aus APS Header und APS Payload. Ersterer beinhaltet das Feld ”Frame Control”, Felder zur Adressierung, einen ”APS Counter” sowie einen optionalen ”Extended Header”, welcher weitere Subfelder enthält. Das ”Frame Control” Feld gibt Auskunft über den Typ, den Sendemodus, das Format der Acknowledgements und die Sicherheitseigenschaften des Frames. Der ”Destination Endpoint” ist das Ziel, an welches das Frame gesendet wird. Dabei kann durch das Feld ”Group Address” gesteuert werden, ob dieser Befehl an eine ganze Gruppe oder einen einzelnen Teilnehmer gerichtet ist. Mittels ”Cluster Identifier” kann festgestellt werden zu welchem Cluster der Frame gehört. Der ”Profile Identifier” dient zur Zuordnung des Frames zu einem ZigBee-Profil. Das Feld ”Source Endpoint” beinhaltet die Absenderadresse des ZigBee-Geräts. Der ”APS Counter” dient als Schutz vor Replay Angriffen. Abschließend beinhaltet die ”Payload” die zu übertragenden Daten.

<table>
<tr><td>Octets: 1</td><td>0/1</td><td>0/2</td><td>0/2</td><td>0/2</td><td>0/1</td><td>1</td><td>0/Variable</td><td>Variable</td></tr>
<tr><td rowspan="2">Frame control</td><td>Destination Endpoint</td><td>Group address</td><td>Cluster identifier</td><td>Profile identifier</td><td>Source endpoint</td><td rowspan="2">APS counter</td><td rowspan="2">Extended Header</td><td rowspan="2">Frame payload</td></tr>
<tr><td colspan="5">Adressing fields</td></tr>
<tr><td colspan="8">APS Header</td><td>APS Payload</td></tr>
</table>

Abbildung 2.5: ZigBee APDU Format [17]

2.3 ZigBee Topologien

In der Spezifikation wird zwischen Endgerät, Router und Koordinator unterschieden, wobei der ZigBee-Koordinator gleichzeitig auch den PAN Koordinator und in weiterer Folge meist das Trust Center darstellt. Dieser ist in jedem zentralisierten Netzwerk zwingend erforderlich und fungiert als zentrale Stelle, welche das Management und die Sicherheit des ZigBee-Netzwerks übernimmt. Router werden auch als Koordinatoren bezeichnet und sind Full-Function Devices (FFD). Einfache Endgeräte, welche über wenige Funktionen und geringen Speicherplatz verfügen sowie keine großen Datenmengen übertragen können werden auch Reduced-Function Devices (RFD) genannt. Ein RFD kann weder PAN Koordinator noch Koordinator sein kann. Ein weiteres Unterscheidungsmerkmal ist, dass ein RFD die Kommunikation nur zu einem anderen RFD aufbauen kann, wohingegen ein FFD Nachrichten an andere FFD’s sowie RFD’s senden kann. Auf Grund der Eigenschaften des zugrundeliegenden 802.15.4 WPAN Standards kann zwischen drei Arten von Topologien unterschieden werden, welche anschließend in Abbildung 2.6 grafisch dargestellt werden [10, 18, 11]:

- Stern

In einem Stern Netzwerk gibt es mit dem PAN Koordinator ein zentrales Gerät, welches das Netzwerk verwaltet. Hierbei handelt es sich um ein mit Strom versorgtes FFD damit eine ständige Verfügbarkeit gewährleistet werden kann. Wenn ein Full-Function Device das erste Mal gestartet wird kann dieses ein eigenes Netzwerk erstellen wobei es hierfür eine freie PAN ID verwendet. Dazu wird der verwendete Frequenzbereich gescannt und auf andere ZigBee-Netzwerke überprüft, um die Eindeutigkeit der PAN ID sicherzustellen. Bei einem Stern Netzwerk kommuniziert jedes Endgerät direkt mit dem PAN Koordinator. Diese Topologie Form wird sehr häufig im Bereich der Heimautomatisierung verwendet.

- Mesh

  Im Unterschied zum Stern-Netzwerk können bei einem Mesh Netzwerk die Geräte auch untereinander kommunizieren. Die Kommunikation muss also nicht zwingend über den PAN Koordinator stattfinden, sondern kann über mehrere ZigBee-Router an das Zielgerät geroutet werden. Dadurch entsteht ein verlässliches Netzwerk mit Multipath Routing Funktionalität. Mesh Netzwerke werden häufig bei Smart Grid Applikationen verwendet.

- Baum

  In einem Baum Netzwerk gibt es ebenso nur einen PAN Koordinator, wobei die meisten der Netzwerkteilnehmer FFDs darstellen. Beim Erstellen des Netzwerks generiert der PAN Koordinator einen Cluster und definiert sich selbst als Cluster Head (CH) mit der Cluster ID null. Beim Zutritt eines neuen Teilnehmers zum Netzwerk fügt der Cluster Head diesen als Child in seine Neighbor Tabelle hinzu. Das beitretende Gerät wiederum ergänzt diesen als Parent in seiner Neighbor Tabelle. Wenn das Netzwerk eine bestimmte Größe erreicht, kann der PAN Koordinator ein oder mehrere weitere FFD's dazu veranlassen einen eigenen Cluster zu bilden und die Rolle des Cluster Heads vom jeweiligen Cluster zu übernehmen. Dies führt zwar zu einer größeren Latenz bei der Übertragung der Nachrichten, hat aber den großen Vorteil, dass auf Grund dieser Clusterstruktur eine weitaus größere Fläche abgedeckt werden kann. Bei einem Baum Netzwerk kommuniziert ein RFD nicht zwingend mit dem PAN Koordinator, sondern mit sämtlichen FFD's in seiner Umgebung. Auf Grund der Möglichkeit sehr viele Endgeräte über große Distanzen hinweg zum Netzwerk hinzufügen zu können wird diese Form der Topologie häufig für die Automatisierung in der Industrie verwendet.

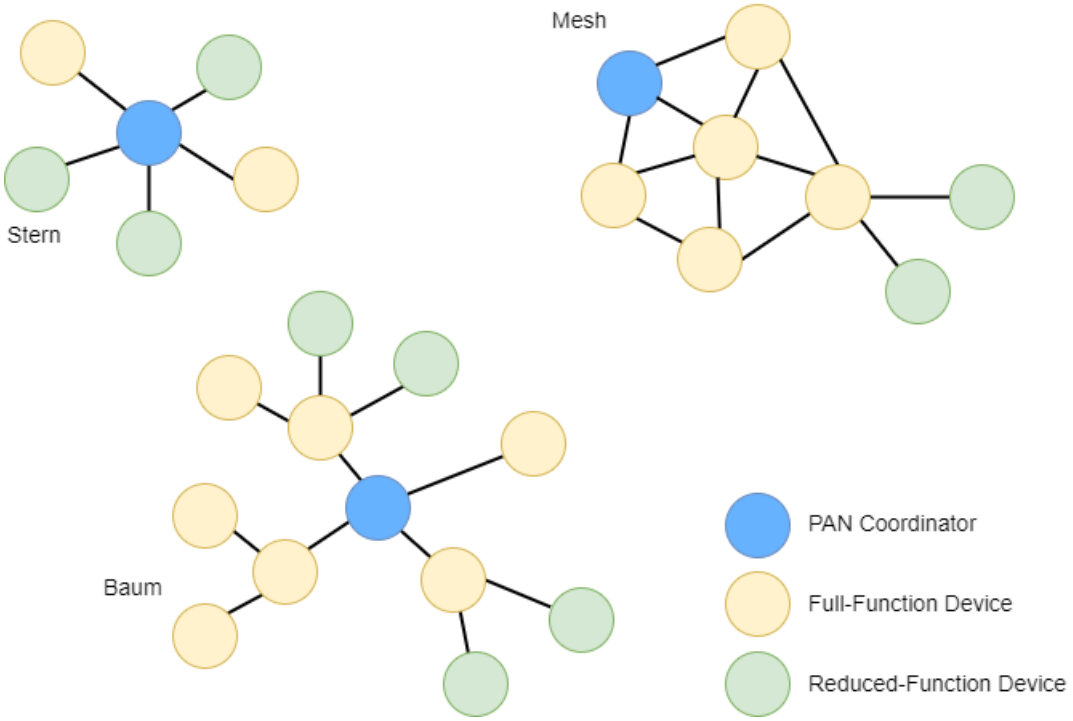

Abbildung 2.6: ZigBee Topologien [10]

2.4 Arten für den Beitritt in ein ZigBee-Netzwerk

Für den sicheren Betrieb eines ZigBee-Netzwerks ist die Geheimhaltung des Netzwerk-Keys sehr wichtig. Dieser wird beim Zutritt vom Trust Center an das beitretende Gerät übermittelt. Um das Beitreten von ungewollten Teilnehmern zu verhindern kann im Trust Center, nachdem alle Teilnehmer mit dem ZigBee-Netzwerk verbunden sind, das ”permit-joining” Attribut deaktiviert werden. Somit können dem Netzwerk keine neuen Teilnehmer mehr beitreten. Aus Sicht des ZigBee Standards gibt es verschiedene Szenarien unter welchen Umständen ein Teilnehmer einem ZigBee-Netzwerk beitreten kann [19, 20]:

2.4.1 Association

Diese Methode beschreibt den Standardvorgang, wenn ein neuer Teilnehmer dem Netzwerk beitreten möchte. Dabei wird vom beitretenden Gerät ein ”Association Request” ausgeschickt und vom Koordinator mit einem ”Association Response” beantwortet. Zweiteres beinhaltet die Antwort, ob der Teilnehmer dem Netzwerk beitreten darf. In einem zentralen Netzwerk stellt das Trust Center den Koordinator dar, wohingegen im Falle eines verteilten Netzwerks ein ZigBee-Router auf den Request antwortet. Diese Request- und Response-Pakete werden in definierten Management Paketen übermittelt, wobei das ”permit-joining” Attribut aktiviert sein muss, damit das ZigBee-Netzwerk neue Verbindungen akzep-

tiert.

2.4.2 Rejoin

Damit ein zuvor bereits verbundenes Gerät einem Netzwerk erneut beitreten kann gibt es die Funktionalität des Rejoins. Dabei treten die Geräte dem Netzwerk über ihren Parent bei, wobei dieser ein ZigBee-Router oder das Trust Center sein kann. Dies funktioniert, auch wenn das Netzwerk für neue Verbindungen geschlossen ist. Somit ist es möglich alle neuen Geräte auszusperren, die bereits bekannten Geräte hingegen können sich jederzeit neu verbinden. Hierbei werden Daten Pakete verwendet, um die Netzwerk Parameter auszutauschen.

2.4.3 Rejoin nach Orphan Scan

Wenn sich ein Endgerät, beispielsweise nach einem Stromausfall, wieder mit seinem Parent verbinden möchte schickt dieses Broadcast Pakete aus um zu prüfen ob dieser noch erreichbar ist. Sollte der Parent zu dem Zeitpunkt nicht erreichbar sein scannt der Teilnehmer das ZigBee-Netzwerk, um diesem erneut beizutreten. Hierbei spricht man von einem Orphan Scan wobei, die im ZigBee Standard definierten, Management Pakete verwendet werden und das Netzwerk nicht für neue Verbindungen offen sein muss.

2.4.4 Out-of-band

Hierbei handelt es sich um Methoden, bei welchen der Netzwerkschlüssel nicht via ZigBee auf die beitretenden Geräte übertragen wird und somit auch nicht mittels Sniffing des ZigBee-Traffics abgehört werden kann. So kann beispielsweise der Netzwerk-Key schon bei der Produktion in das Endgerät integriert werden. Dadurch wird sichergestellt, dass dieser niemals übertragen werden muss und somit auch nicht abgefangen werden kann. Ein weiterer Vorteil ist, dass Rejoins hierbei immer verschlüsselt stattfinden, da der Netzwerk-Key zum Zeitpunkt des Betritts zum ZigBee-Netzwerk bereits auf dem Gerät vorhanden ist. Eine weitere Möglichkeit ist die Speicherung des Schlüsselmaterials auf einem NFC-Chip wodurch die Reichweite, in welcher der Key ausgelesen werden könnte, drastisch reduziert wird. Nichtsdestotrotz könnte man in diesem Fall den Schlüssel einfach mittels NFC Reader auslesen. Auch das Kodieren der Daten für die Inbetriebnahme mittels QR-Code oder Barcode ist möglich. Das hat wiederum den Nachteil, dass es sehr einfach mittels QR-Code Scanner ausgelesen werden kann. Eine weitere Möglichkeit ist, dass das Gerät die Schlüssel online von einer Webseite bezieht. Dazu muss das Gerät auf der Webseite registriert werden und braucht in weiterer Folge eine intakte Internetverbindung, um die Schlüssel beziehen zu können.

2.5 Unterschiede im Datentransfer

ZigBee-Geräte können auch anhand deren Stromversorgung und den dadurch resultierenden Möglichkeiten des Datentransfers unterschieden werden. So können direkt mit Strom versorgte Geräte jederzeit Befehle entgegennehmen. Hierbei kann der Koordinator die Verbindung zum Endgerät aufbauen und dieses verarbeitet den übertragenen Befehl. Im Gegensatz dazu gibt es batteriebetriebene Geräte, welche sich aus Gründen der effizienten Nutzung des zur Verfügung stehenden Stroms meist im Ruhemodus befinden und periodisch selbst eine Verbindung zum Koordinator initiieren, um neue Befehle abzuholen. Dieser Unterschied wird in Abbildung 2.7 grafisch dargestellt. [21]

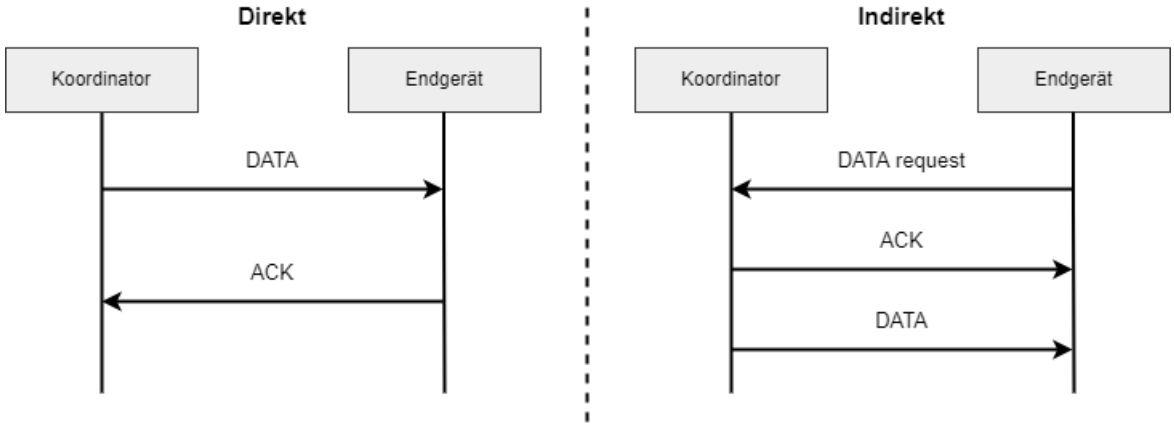

Abbildung 2.7: Unterschiede im Datentransfer [21]

2.6 ZigBee Cluster Library (ZCL)

ZigBee-Cluster sowie die in Abschnitt 2.7 beschriebenen Application Profiles werden verwendet, um die Interoperabilität zwischen Produkten von verschiedenen Herstellern zu gewährleisten, wobei erstere nach dem Client/Server Prinzip funktionieren. Beispielsweise wird das, im Applikationsprofil Smart Energy verwendete, zertifikatsbasierte Verfahren zum Schlüsseltausch im ”Key Establishment” Cluster beschrieben. [22]

Mittels ZCL Frames kann ein Client bestimmte Befehle an einen Server übertragen, um bestimmte Attribute zu verändern. Ein Server ist beispielsweise eine Lampe, wobei die zugehörigen Befehle An, Aus und Umschalten sein können. Nach dem Erhalt des jeweiligen ZCL Befehls wird die Lampe ihren Zustand verändern. In der ZigBee Cluster Library werden alle verschiedenen ZigBee-Cluster zusammengefasst. [1]

2.7 ZigBee Application Profiles

ZigBee Applikationsprofile dienen als Grundlage für die Kommunikation in ZigBee-Netzwerken. Dabei beschreiben diese Profile Richtlinien wie Nachrichten aussehen und verarbeitet werden müssen. Somit kann sichergestellt werden, dass Applikationen herstellerunabhängig miteinander kommunizieren können, sofern diese dasselbe Applikationsprofil verwenden. Sehr weit verbreitete Profile sind beispielsweise ZigBee Light Link, ZigBee Home Automation sowie Smart Energy.

2.7.1 ZigBee Light Link Profile (ZLL)

Das ZigBee Light Link Profil dient zur Steuerung einer Vielfalt von Lampen und anderen Lichtquellen. Unter anderem kann dieses für die Steuerung der Helligkeit sowie das Ein- und Ausschalten verwendet werden. Aus Gründen der Einfachheit wird bei ZLL auf ein Trust Center verzichtet, wodurch sich das Schlüsselmanagement auf einen Netzwerkschlüssel, welcher dem Gerät beim Beitritt ins Netzwerk übermittelt wird, begrenzt. Diese initiale Übertragung des Netzwerkschlüssels erfolgt verschlüsselt, wobei hierzu ein Default Trust Center Link Key verwendet wird. Dieser Default Key wurde an alle zertifizierten Hersteller verteilt um herstellerübergreifend den Zutritt zu einem ZigBee-Netzwerk zu ermöglich. Nachdem das beitretende Gerät im Besitz des Netzwerk-Keys ist wird die Kommunikation auf Netzwerk-Layer Ebene mit diesem gesichert. ZLL bietet mit ”Touchlink Commissioning” ein Feature zur einfachen Integration von Geräten ins Netzwerk, wobei keine Interaktion mit dem/der Benutzer*in notwendig ist. Wenn ein ZigBee-Gerät mittels Touchlink Commissioning zu einem Netzwerk hinzugefügt wird werden mehrere Schritte durchlaufen, welche nachstehend in Abbildung 2.8 abgebildet werden. [23]:

1. Scan Request: Broadcast an alle Touchlink fähigen Geräte
2. Scan Response: Ist der Router/das Endgerät bereit dem Netzwerk des Initiators beizutreten antwortet dieses mit einem Response
3. Device Informationen Request: Optional können auch Informationen des ZigBee-Gerätes abgefragt werden
4. Identify Request: Optional kann ein Request zur Identifizierung an das Zielgerät gesendet werden, wodurch beispielsweise die jeweilige Lampe einmal blinkt
5. Network Join Request: Dieser Request beinhaltet den verschlüsselten Netzwerk-Key
6. Network Join Response: Das Zielgerät sendet einen Response an den Initiator mit der Information ob dieses dem Netzwerk erfolgreich beigetreten ist

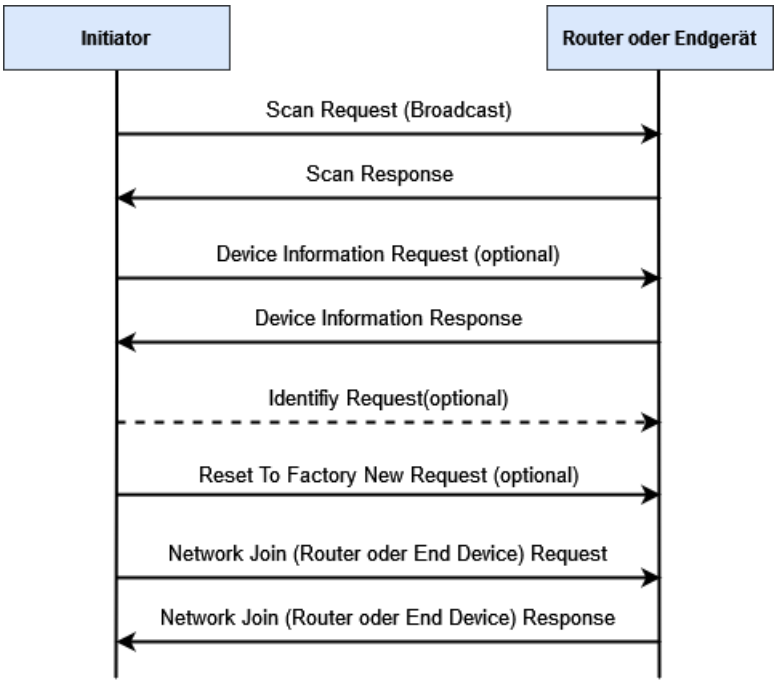

Abbildung 2.8: Touchlink Commissioning [23]

Der Beitritt zum Netzwerk wird dabei lediglich durch den bereits öffentlich bekannten Link Key (”ZigBeeAlliance09”) sowie der geringen Distanz zwischen beitretendem Gerät und Initiator gesichert, wodurch dies auch für Angreifer*innen interessant ist. Eine detaillierte Beschreibung des Angriffs sowie der Erkennung durch das NIDS (Network Intrusion Detection System) folgt in Abschnitt 4.2 sowie Unterabschnitt 5.5.3. [21]

2.7.2 ZigBee Home Automation Profile

Das ZigBee Home Automation Profil kann zur Steuerung sämtlicher kompatiblen Geräte verwendet werden und ist vor allem in der Industrie sowie dem Smart Home Bereich weit verbreitet. Hierzu zählen Waschmaschinen, Kühlschränke, Unterhaltungselektronik wie 3D Brillen, Funkmelder oder Kaffeemaschinen. Wie auch schon beim Light Link Profil wird die Kommunikation mittels Netzwerkschlüssel gesichert. Im Gegensatz zum ZLL Profil wird hier aber ein Trust Center verwendet, welches sich um das Schlüsselmanagement und den Zutritt zum Netzwerk kümmert. Um die Interoperabilität der Geräte herstellerübergreifend gewährleisten zu können ist in diesem Profil in den Startup Attribut Sets (SAS) ein Default Key hinterlegt mit welchem der Netzwerkschlüssel gesichert zu beitretenden Geräten übertragen werden kann. Dieser Default Trust Center Link Key ist jedoch mittlerweile ebenso öffentlich bekannt und hat den Wert „5A 69 67 42 65 65 41 6C 6C 69 61 6E 63 65 30 39“ (ZigBeeAlliance09). Da die Sicherheit des Netzwerks vom Netzwerk-Key abhängig ist und dieser bei der initialen Übertragung nur

mit dem bereits bekannten Default Key verschlüsselt wird stellt dies ein hohes Sicherheitsrisiko dar. Ob dieser Default Link Key als Fallback verwendet werden soll, ist als Attribut namens „Default Link Key Join“ - 0x01 (True) - ebenso in den Startup Attribut Sets hinterlegt. Da im Gegensatz zum Light Link Profil meist keine Lichtanlagen sondern auch Türschlösser gesteuert werden, können die Auswirkungen dieses Risikos viel größer sein. [21, 1]

2.7.3 Smart Energy Profile (SEP)

Das Anwendungsgebiet vom Smart Energy Profil erstreckt sich von der Messung des Zählerstandes, über die Lastensteuerung bis hin zur effizienten Nutzung von Energie. Dabei kann beispielsweise überschüssige Energie aus der Photovoltaik Anlage direkt dazu verwendet werden, um den Warmwasserspeicher aufzuheizen. Durch die Verwendung des Smart Energy Profils ist es somit durch effizientes Energiemanagement möglich Kosten zu senken. [24]
Die Kryptografie beim Smart Energy Profil basiert auf PKI (Public Key Infrastructure) wobei zur Authentifizierung und als Schlüsseltauschverfahren, wie in Unterabschnitt 3.6.3 beschrieben, ein zertifikatsbasiertes Verfahren verwendet wird. Dies ist eine sehr sichere Variante zum Austausch des Schlüsselmaterials. Das Problem hierbei ist, dass solch ein Zertifikat von jedem ausgestellt werden kann. Die Voraussetzung ist lediglich, dass die Geräte dem Aussteller dieses Zertifikates vertrauen. Um nun Interoperabilität zwischen den Herstellern zu gewährleisten, müssten alle Geräte den CA’s aller Hersteller vertrauen. Dies ist jedoch auf Grund der geringen Speicherkapazität auf den Endgeräten nicht möglich. So kann es passieren, dass Geräte einem gültigen Zertifikat nicht vertrauen, weil der jeweilige Aussteller nicht im Trust Store des Endgeräts gespeichert ist. [12]

3 Sicherheit von ZigBee

Bereits im Jahr 2008 wurden im ZigBee Standard einige Sicherheitsmaßnahmen spezifiziert, welche im Netzwerk- sowie dem Applikations-Layer angesiedelt sind. Hierzu gehören die Erstellung und Verteilung der Schlüssel sowie die Sicherheit der Geräte und der Frames. Diese Schutzziele sind ebenfalls im aktuellen Standard aus dem Jahr 2015 noch enthalten. Weiters basiert ZigBee auf dem ”Open Trust” Modell wodurch sich alle Protokollschichten gegenseitig vertrauen. Somit müssen die Sicherheitsmaßnahmen nur bei Kommunikation mit anderen Geräten implementiert werden, wobei jeder Layer für die Sicherheit der eigenen Frames verantwortlich ist. Weiters ermöglicht dies, dass Schlüssel auf verschiedenen Layern auf einem Gerät wiederverwendet werden können. In einem ZigBee-Netzwerk verwenden alle Geräte und deren Layer dasselbe Sicherheitslevel, um eine bessere Interoperabilität gewährleisten zu können. [25, 17, 21, 26]
Da es seitens des Standards aber keine zwingenden Sicherheitsrichtlinien gibt liegt die Verantwortung für die Umsetzung hierbei bei den Herstellern. Bei ZigBee kann die Sicherheit in technische und organisatorische Maßnahmen unterteilt werden [2, 19]:

- Organisatorisch - können die Benutzer*innen im Trust Center konfigurieren
  - Sicherheitslevel
  - Blacklist und Whitelist
  - Berechtigungen im Netzwerk
  - ACL (Access Control Lists)
- Technisch
  - Schlüsselmanagement
  - Verschlüsselung
  - Authentifizierung
  - Integrität
  - Frame Counter

3.1 Organisatorische Sicherheitsmaßnahmen

Mittels Trust Center ist es möglich die Sicherheit des ZigBee-Netzwerks zu erhöhen. Dabei bieten sich verschiedene sicherheitsrelevante Konfigurationsmöglichkeiten an. [19]

3.1.1 Sicherheitslevel

Das Trust Center ist für die Auswahl des entsprechenden Sicherheitslevels zuständig. Dabei wird zwischen Standard-Security und High-Security unterschieden, wobei der Unterschied zum großen Teil das Management und das Verteilen der Schlüssel darstellt. Im Falle von Standard-Security wird der Netzwerk-Key unverschlüsselt übertragen. Bei High-Security hingegen wird dieser mit dem Master-Key verschlüsselt übertragen.

3.1.2 Blacklists und Whitelists

Es ist möglich mittels Whitelist nur definierte Geräte anhand deren MAC Adressen im Netzwerk zu erlauben. Im Gegensatz dazu ist es auch möglich mittels Blacklist bestimmte MAC Adressen zu sperren, wobei diese Listen im Trust Center zu pflegen sind. Aus der Sicherheitsperspektive ist es ratsam eine Whitelist zu verwenden und diese beim Kauf neuer Geräte, um deren MAC Adressen zu erweitern. Das effiziente Führen einer Blacklist ist nur schwer möglich, da man die MAC Adresse eines/einer potenziellen Angreifers/Angreiferin vorab nicht kennt. [19]

3.1.3 Berechtigungen im Netzwerk

Es ist möglich den Geräten im Trust Center bestimmte Rechte zuzuordnen. Somit können die Rechte der ZigBee-Geräte im Netzwerk auf ein Minimum beschränkt werden. Exemplarisch seien hier zwei Berechtigungen aufgelistet [19]:

- Das Recht einen Link-Key anzufordern, um mit einem anderen Gerät im Netzwerk kommunizieren zu können. Hierfür wird das ”request key” Kommando verwendet.
- Das Recht einem Netzwerk mittels Rejoin erneut beizutreten. Hiermit ist es möglich den Angriff des Insecure Rejoins, welcher in Abschnitt 4.1 beschrieben wird, zu unterbinden.

3.1.4 Access Control List (ACL)

Die ACL kann verwendet werden, um unautorisierten Geräten den Zugriff auf Ressourcen innerhalb des Netzwerks zu verweigern. Die ACL wird in der MAC PAN Information Base (PIB) eines Gerätes

gespeichert und beinhaltet sicherheitsrelevante Informationen wie Sicherheitslevel, Schlüssel und Frame Counter. [27]

3.2 Technische Sicherheitsmaßnahmen

In diesem Kapitel sind technische Maßnahmen zusammengefasst, auf welche die Benutzer*innen meist keinen oder nur wenig Einfluss haben.

3.2.1 Schlüsselmanagement

Wie bereits erwähnt ist das Trust Center für das Schlüsselmanagement zuständig, wobei man grundlegend zwischen drei Arten von Schlüsseln unterscheidet:

- Link-Key
- Network-Key
- Master-Key

Die Unterschiede zwischen diesen sowie die Verfahren zum Austausch dieser werden in Abschnitt 3.5 und Abschnitt 3.6 detailliert beschrieben.

3.2.2 Verschlüsselung

Bei ZigBee wird das symmetrische Verschlüsselungsverfahren AES-CCM (Advanced Encryption Standard und Counter mit CBC-MAC) mit einer Schlüssellänge von 128-Bit verwendet. Hierbei teilen sich die Kommunikationsteilnehmer einen gemeinsamen privaten Schlüssel. Durch die Verwendung dieses Verfahrens wird die Authentifizierung, Vertraulichkeit und Integrität sichergestellt. [19, 11]
AES-CTR beschreibt den Algorithmus zur Ver- und Entschlüsselung der Daten, wobei diese dabei in Blöcke unterteilt und verschlüsselt werden. Dieser häufig verwendete Verschlüsselungsalgorithmus wird in der Arbeit von Lipmaa [28] sehr genau beschrieben.
Mittels CBC-MAC (cipher block chaining message authentication code) wird die Integrität und Authenzität der Nachricht sichergestellt. Dabei wird die Nachricht in Blöcke unterteilt und der erste Klartextblock anschließend mit dem privaten Schlüssel verschlüsselt. Der so entstehende Block wird mit dem nächsten Klartextblock XOR verknüpft und danach wiederum dem Verschlüsselungsalgorithmus zugeführt. Eine detaillierte Beschreibung dieses Verfahrens fand bereits in anderen Arbeiten [29] statt.

3.2.3 Authentifizierung

Die Authentifizierung wird auf Netz- und Applikations-Layer Ebene mittels Netzwerk-Key sowie den Link-Keys gewährleistet. Dadurch können Informationen zwischen den Geräten synchronisiert werden, wobei die Authentizität durch die gemeinsamen Schlüssel sichergestellt wird. [15]

3.2.4 Integrität

Um sicherstellen zu können, dass die Pakete bei der Übertragung nicht verändert wurden, wird ein MIC (Message Integrity Code) verwendet, welcher mittels CCM generiert wird. Hierbei werden 16-Bit, 32-Bit, 64-Bit sowie 128-Bit MIC vom ZigBee Standard unterstützt. [15]

3.2.5 Frame Counter

Der Frame Counter (FC) stellt einen 32-bit großen numerischen Wert dar, welcher bei jedem neuen Paket inkrementiert wird. Ein ZigBee-Gerät prüft während der Kommunikation im ersten Schritt immer die Gültigkeit des Frame Counters bevor es ein Paket akzeptiert. Dabei werden alle Pakete verworfen, bei denen der FC niedriger oder gleich dem vom vorherigen Paket ist. Somit können Replay Angriffe unterbunden werden, da bei diesen aufgezeichnete Pakete erneut gesendet werden. Da in diesem Replay Paket der Frame Counter nicht höher ist, als der letzte den das ZigBee-Gerät erhalten hat, verwirft dieses das eingehende Paket und führt somit den Befehl nicht aus. [11, 15]

3.3 ZigBee Sicherheitsmodelle

In ZigBee gibt es zwei Arten von Sicherheitsmodellen, wobei zwischen zentralem und verteiltem Modell unterschieden wird. Der Unterschied liegt im Vorhandensein einer zentralen Stelle namens Trust Center, welches sich um das Management des Netzwerks kümmert. Bei beiden Modellen beruht die Sicherheit auf der Geheimhaltung der Link-Keys und des Netzwerk-Keys. [11]

3.3.1 Zentrales Sicherheitsmodell

Bei einem zentral verwalteten Netzwerk ist das Trust Center für die Kontrolle über das Netzwerk zuständig. Hierbei muss sich jeder Teilnehmer beim Netzwerkbeitritt gegen dieses authentifizieren. Ebenso kümmert sich alleinig das Trust Center um das Schlüsselmanagement und die Verteilung der Schlüssel. Hierbei verwenden die Geräte vom Trust Center ausgestellte Link-Keys für die Unicast Kommunikation. Ein Vorteil dieser weit verbreiteten Lösung ist, dass das Trust Center alle Nodes kennt und somit einen Überblick über das gesamte ZigBee-Netzwerk bietet. [11, 26]

3.3.2 Verteiltes Sicherheitsmodell

Seit ZigBee 3.0 ist es auch möglich innerhalb eines ZigBee-Netzwerks ein dezentrales Sicherheitsmodell zu betreiben. Hierbei besteht das ZigBee-Netzwerk lediglich aus einem oder mehreren Routern und Endgeräten. Dabei wird die Verwaltung des Netzwerks auf sämtliche ZigBee-Router aufgeteilt, wobei sich die neuen Teilnehmer gegen einen von diesen authentifizieren müssen. Auf Grund dessen ist hierbei auch die Aufgabe der Schlüsselverteilung auf alle ZigBee-Router verteilt, wodurch es in diesem Modell kein zentrales Gerät gibt. Auf Grund der Absenz eines Trust Centers wird das System zwar einfacher, ist aber zugleich auch unsicherer. Bei dem verteilten Sicherheitsmodell wird sämtlicher Traffic mit dem Netzwerkschlüssel verschlüsselt. [19, 11, 26] In Abbildung 3.1 werden die beiden Arten grafisch dargestellt.

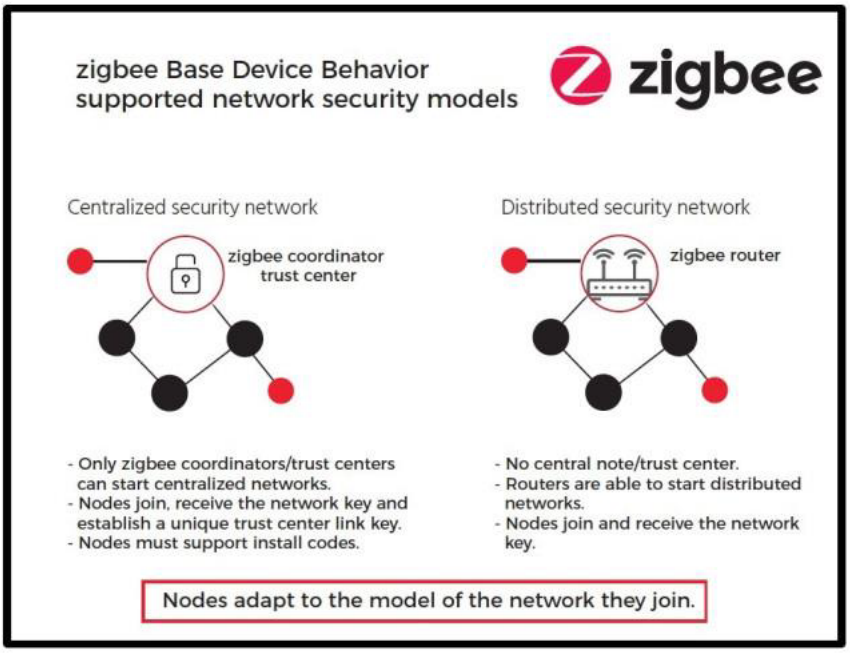

Abbildung 3.1: ZigBee Sicherheitsmodelle [11]

3.4 Grundvoraussetzung für sichere Netzwerke

Neben sehr vielen konfigurierbaren Sicherheitsmaßnahmen hängt die Sicherheit von ZigBee-Netzwerken sehr stark von folgenden grundlegenden Punkten ab [26]:

- Geheimhaltung der privaten Schlüssel

  Einer der wichtigsten Punkte bei der Sicherheit von ZigBee-Netzwerken ist die Geheimhaltung der Schlüssel. Dabei setzt ZigBee voraus, dass Schlüsselinformationen niemals unverschlüsselt über-

tragen werden. Als einzige Ausnahme sei die Vorkonfiguration von einem neuen Gerät, bei welcher ein Key unverschlüsselt übertragen werden könnte, genannt. Auf Grund der generellen Konzeption als low-cost Komponenten sieht die ZigBee Security Policy keinen Schutz vor Angriffen auf die Hardware vor.

- Schutz des verwendeten Modells
  ZigBee-Geräte müssen das zentrale sowie dezentrale Sicherheitsmodell unterstützen. Im Zuge des Beitritts zu einem Netzwerk muss das beitretende Gerät das Sicherheitsmodell des Netzwerks übernehmen.

- Sichere Implementierung von kryptografischen Verfahren sowie der zugehörigen Policies
  Der ZigBee Standard geht davon aus, dass sich die Entwickler an die vorgegebenen Richtlinien und Verfahren halten. Weiters wird davon ausgegangen, dass gute Zufallszahlengeneratoren verwendet werden.

3.5 ZigBee-Keys

Auf Grund der Verwendung eines symmetrischen Verschlüsselungsverfahrens, welches in Unterabschnitt 3.2.2 detailliert beschrieben wird, basiert die Sicherheit von ZigBee-Kommunikation auf der Geheimhaltung der verwendeten privaten 128-Bit Schlüssel. Hierbei kann prinzipiell zwischen drei Arten von Schlüsseln unterschieden werden [25]:

- Netzwerk-Key
- Link-Keys
- Master-Keys

In Abbildung 3.2 werden die Zuständigkeiten der verschiedenen Schlüssel exemplarisch dargestellt. Dieser kann entnommen werden, dass auf Netzwerk-Layer (NWK) Ebene alle Geräte innerhalb des ZigBee-Netzwerks denselben Schlüssel verwenden. Die Link-Keys, mit denen der Applikations-Layer Traffic verschlüsselt wird, teilen sich jeweils nur die miteinander kommunizierenden Teilnehmer. So gibt es beispielsweise für die Kommunikation zwischen Trust Center und Lampe einen eigenen Link-Key, den die anderen Teilnehmer nicht kennen. Jedes Gerät hat immer nur die Link-Keys, welche zur eigenen Kommunikation mit anderen Teilnehmern verwendet werden, gespeichert. Die Verteilung der Keys übernimmt in einem zentralen Netzwerk das Trust Center, wie in Abschnitt 3.6 detailliert erklärt wird.

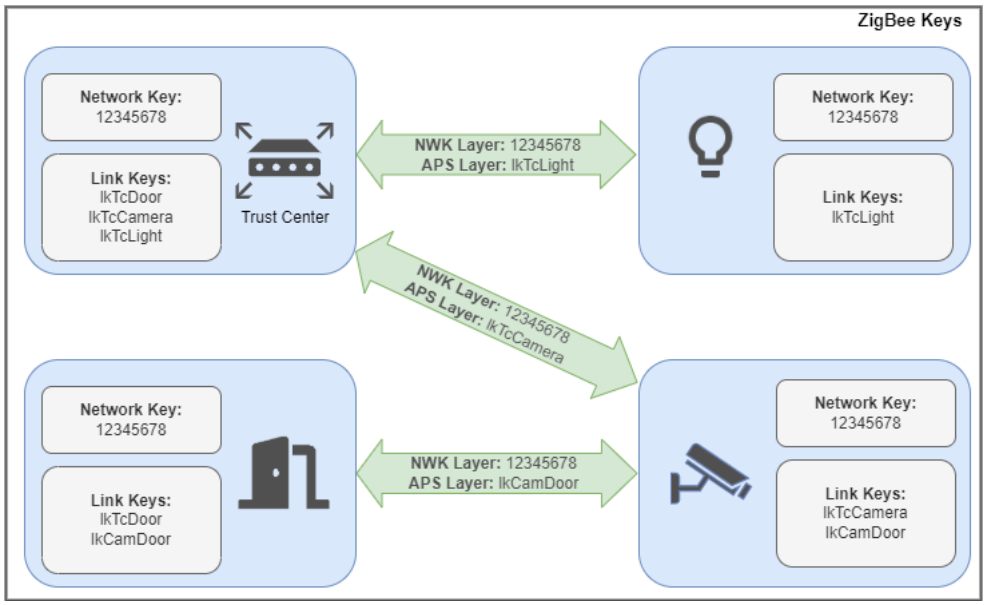

Abbildung 3.2: ZigBee-Keys

3.5.1 Netzwerk-Key

In einem ZigBee-Netzwerk verwenden alle Teilnehmer den gleichen Netzwerk-Key, um die Kommunikation auf Netzwerk-Layer Ebene zu verschlüsseln. Hierbei handelt es sich vor allem um Nachrichten, welche für das Management oder die Kontrolle des Netzwerks verwendet werden. In einem Trust Center können zwar mehrere Netzwerkschlüssel gespeichert sein, wobei der aktive Schlüssel anhand der Sequenznummer erkannt werden kann. Dieser wird, wie in Abschnitt 3.6 beschrieben, entweder im Zuge des Beitritts zum Netzwerk auf das Gerät übertragen oder vorab auf diesem installiert. Weiters kann der Netzwerk-Key auch in der Verschlüsselung auf Applikations-Layer Ebene Anwendung finden. Dies ist möglich wenn entweder kein Link-Key bezogen werden kann oder der Netzwerk-Layer explizit dazu aufgefordert wird die APL-Frames zu verschlüsseln. [21, 19]
Bei der Verteilung wird der Netzwerkschlüssel mit einem Link-Key geschützt. In einem ZigBee-Netzwerk, welches das zentrale Sicherheitsmodell verwendet, ist dieser Link-Key dem Trust Center sowie dem beitretenden Gerät bekannt. Beim verteilten Sicherheitsmodell ist dieser Link-Key allen Teilnehmern bekannt. [26]

Rotation des Netzwerk-Keys

Aus Gründen der Sicherheit sollte der Netzwerk-Key regelmäßig geändert werden. Die Vorgehensweise für die Rotation des Netzwerkschlüssels ist im Standard wie folgt definiert [19]:

1. Das Trust Center schickt per Broadcast einen neuen Netzwerk-Key, welcher mit dem Trust Center Link Key verschlüsselt ist, an alle Teilnehmer im ZigBee-Netzwerk.

2. Das Trust Center schickt den Befehl ”switch-key” an alle Teilnehmer, wodurch ab sofort der neue Key gültig ist. Ein ZigBee-Gerät kann mehrere Netzwerk-Keys speichern, wobei es den aktiven Key immer anhand einer eindeutigen ”key sequence number” identifiziert.

3. Ab sofort verschlüsseln alle Teilnehmer die Kommunikation auf Netzwerk-Layer Ebene mit dem neuen Schlüssel.

4. Es besteht die Möglichkeit, dass zum Beispiel batteriebetriebene Geräte den Wechsel des Netzwerkschlüssels verpassen, weil sie sich zu diesem Zeitpunkt im Ruhemodus befinden. In diesem Fall wird beim nächsten Verbindungsversuch der neue Netzwerk-Key, mit dem Transport-Key verschlüsselt, übertragen.

Im ZigBee Standard ist definiert, dass der Netzwerk-Key periodisch ausgetauscht werden sollte. Leider wird hierbei aber kein Intervall definiert, in welchem die Rotation des Netzwerk-Keys stattfinden sollte. Aus diesem Grund wird dieser sicherheitsrelevante Aspekt auch in Unterabschnitt 3.6.4 detailliert behandelt. [12]

3.5.2 Link-Key

Mit Link-Keys wird die Unicast Kommunikation zwischen zwei Teilnehmern auf Applikations-Layer Ebene verschlüsselt. Diese privaten Schlüssel dienen als zusätzliche Sicherheitsmaßnahme und sind nur den beiden miteinander kommunizierenden Teilnehmern bekannt. Grundlegend wird hierbei zwischen ”Global Link-Keys” und ”Unique Link-Keys” unterschieden, wobei der Unterschied darin liegt wie das Gerät mit den APS Befehlen vom Trust Center umgeht. [26] Der Link-Key, welcher zur Verschlüsselung der Nachrichten zwischen einem Endgerät und dem Trust Center dient wird auch als Trust Center Link Key (TCLK) bezeichnet. Der TCLK schützt neben den Nachrichten auf Applikations-Layer Ebene auch die Befehle für den ZigBee-Stack. Wenn zwei Geräte kommunizieren werden also die Daten im Netzwerk-Layer mit dem Netzwerk-Key und zusätzlich die Daten im Applikations-Layer mit dem Link-Key verschlüsselt. Das Trust Center generiert diese zufälligen Link-Keys für die Kommunikation der beiden Teilnehmer und kümmert sich ebenso um die Verteilung dieser. Dieser Prozess kann in folgende Schritte gegliedert werden [19, 12]:

1. Jenes Gerät, welches eine Kommunikation zu einem anderen Teilnehmer aufbauen möchte, sendet einen Request an das Trust Center. Dieser beinhaltet die Aufforderung zur Generierung eines Link-Keys für die beiden Teilnehmer.

2. Das Trust Center prüft im ersten Schritt ob das Gerät berechtigt ist solch einen Request zu senden.

3. Ist dies der Fall erstellt das Trust Center den Link-Key und übermittelt diesen an die beiden Teilnehmer. Diese Verbindung ist mittels Netzwerk-Key sowie den jeweiligen Link-Keys zwischen Trust Center und den Teilnehmern verschlüsselt.

Ein weiteres Unterscheidungsmerkmal bei Link-Keys bildet das Sicherheitsmodell. So gibt es je nach Sicherheitsmodell verschiedene Arten, welche folgend beschrieben werden [26]:

Link-Keys beim zentralen Sicherheitsmodell

- Vorkonfigurierter Global Link Key

  Dieser vorkonfigurierte Link-Key wird auf Netzwerk-Layer Ebene verwendet, um die initiale Übertragung des Netzwerkschlüssels abzusichern. Dieser Link-Key ist für alle Nodes im Netzwerk gleich und ist entweder der von ZigBee definierte Key ”ZigbeeAlliance09” oder ein herstellerspezifischer Code. Nachdem das Sicherheitslevel des Netzwerks festgelegt ist, kann dieser des Weiteren auf Application-Layer Ebene verwendet werden, um die Kommunikation zwischen dem Trust Center und allen Geräten zu verschlüsseln.

- Vorkonfigurierter Unique Link Key

  Seit ZigBee 3.0 ersetzt dieser weitgehend den vorkonfigurierten Global Link Key. Der Unterschied ist hierbei, dass dieser immer nur einem Gerät und dem Trust Center bekannt ist. Der vorkonfigurierte Unique Link-Key wird ebenfalls für die initiale Übertragung des Netzwerkschlüssels verwendet. Dieser wird beim Herstellungsprozess beispielsweise in Form von Install Codes in das Gerät integriert. In ZigBee 3.0 besteht der Install Code aus einer zufälligen 128-Bit Nummer, welche mit einem 16-Bit CRC geschützt ist. Dieser Link-Key wird ebenso auf APL Ebene für die Verschlüsselung zwischen Trust Center und dem Gerät verwendet.

- Trust Center Link Key (TCLK)

  Der Trust Center Link Key dient zur Verschlüsselung auf Application-Layer Ebene zwischen einem Teilnehmer und dem Trust Center. Dabei wird dieser entweder vom Trust Center generiert oder vom vorkonfigurierten Unique Link-Key mittel Matyas-Meyer-Oseas (MMO) Hash Funktion abgeleitet. Der TCLK wird vom Trust Center mit dem Netzwerkschlüssel verschlüsselt an den Teilnehmer übertragen. Des Weiteren wird diese Kommunikation zusätzlich mit dem vorkonfigurierten Unique Link-Key geschützt, falls dieser vorhanden ist. Danach verwenden der Teilnehmer und das Trust Center nur noch den Trust Center Link Key anstatt des vorkonfigurierten Link-Key.

Der vorkonfigurierte Unique Link-Key bleibt lediglich gespeichert, falls das Gerät dem Netzwerk zu einem späteren Zeitpunkt erneut beitreten möchte.

- Application Link-Key

  Der Application Link-Key wird verwendet, um die Kommunikation auf Application-Layer Ebene zwischen zwei Teilnehmern zu verschlüsseln, wobei keiner der beiden Teilnehmer das Trust Center darstellt. Wie in Unterabschnitt 3.5.2 detailliert beschrieben fordert einer der beiden Teilnehmer diesen beim Trust Center an, woraufhin dieses einen zufälligen Key generiert und an die beiden Teilnehmer verteilt. Diese Verteilung ist wiederum mit dem Netzwerkschlüssel und falls vorhanden dem vorkonfigurierten Unique Link-Key geschützt. Somit wird sichergestellt, dass nur die entsprechenden Teilnehmer den generierten Link-Key entschlüsseln können, um ihre künftige Kommunikation damit zu sichern.

Link-Keys beim verteilten Sicherheitsmodell

- Distributed Security Global Link Key

  Dieser Key wird bei der Herstellung auf allen Geräten hinterlegt und wird verwendet, um die Kommunikation zwischen dem Parent Router und dem beitretenden Gerät zu verschlüsseln.

- Vorkonfigurierter Link-Key

  Dieser Key wird ebenso beim Herstellungsprozess in das Gerät integriert und zur Verschlüsselung zwischen Parent Router und dem beitretenden Gerät verwendet. Dieser lässt sich in drei Kategorien unterteilen:

  - Development-Key

    Dieser Key wird lediglich in der Phase der Entwicklung, also bevor das Gerät von ZigBee zertifiziert ist, verwendet.

  - Master-Key

    Nach der Zertifizierung verwendet man ausschließlich den Master-Key.

  - Certification-Key

    Dieser Key wird während einer ZigBee Zertifizierung verwendet.

3.5.3 Master-Key

Der Master-Key wird wie bereits erwähnt entweder beim Herstellungsprozess, via key-transport Kommando oder per Userinteraktion mittels PIN oder Passwort vom Trust Center auf das Gerät übertragen.

Dieser wird verwendet, um die Übertragung der Link-Keys zwischen zwei Geräten im APS-Layer zu verschlüsseln. [30]

3.6 Schlüsseltausch

Wie in Abschnitt 3.5 erläutert, gibt es pro ZigBee-Netzwerk nur einen Netzwerk-Key, welcher sämtliche Pakete zwischen den Teilnehmern auf Netzwerk Ebene verschlüsselt. Um die Geheimhaltung dieses Schlüssels gewährleisten zu können wird dieser im Falle eines zentralen Netzwerks beim Zutritt eines Teilnehmers vom Trust Center verschlüsselt an diesen gesendet. Diese initiale Übertragung kann auf verschiedene Arten gesichert werden [19]:

- Vorkonfigurierter Link-Key
- Install Codes
- Certificate-based Key Establishment

3.6.1 Vorkonfigurierter Link-Key

Beim Home Automation Profil wird die initiale Übertragung des Netzwerk-Keys mit einem vorinstallierten Link-Key verschlüsselt. Dieser Link-Key ist allen Herstellern bekannt, damit der sichere Austausch des Netzwerk-Keys auch herstellerübergreifend funktioniert. Des Weiteren wird dieser auch Default Trust Center Link Key (DTCLK) genannt und ist 16-Byte lang. Das Problem hierbei ist jedoch, dass der DTCLK mittlerweile öffentlich zugänglich ist, wodurch die initiale Übertragung entschlüsselt und der Netzwerk-Key extrahiert werden kann. Die neueste Version des Standards namens ZigBee 3.0, verwendet zwar keine Applikationsprofile mehr, unterstützt diese jedoch aus Gründen der Abwärtskompatibilität. In Abbildung 3.3 wird der Ablauf des Zutritts eines Teilnehmers zu einem Netzwerk mittels vorkonfiguriertem Default Trust Center Link Key veranschaulicht. Da nicht gewährleistet werden kann, dass die initiale Übertragung des Netzwerkschlüssels nicht von einem/einer Angreifer*in aufgezeichnet wurde, gilt hier die Sicherheit nur als limitiert.

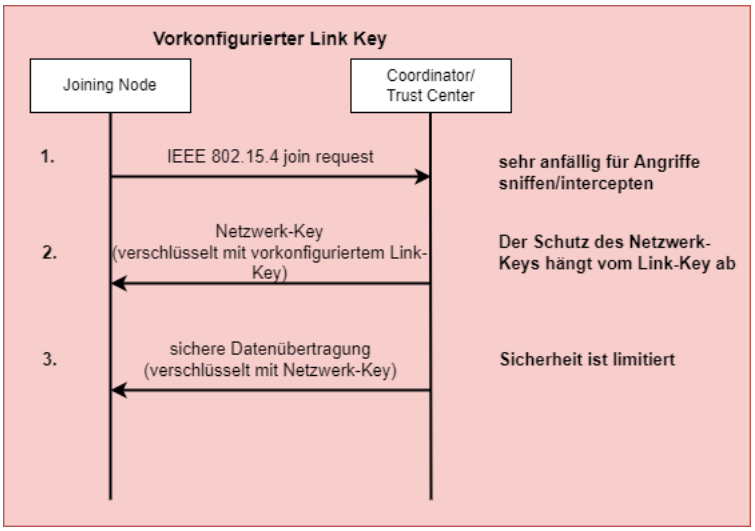

Abbildung 3.3: Vorkonfigurierter Link-Key [19]

Der zum Austausch des Netzwerk-Keys verwendete Link-Key wird auch Transport-Key genannt und wird vom Trust Center Link Key abgeleitet. Der folgenden Abbildung 3.4 kann man die Ableitung des Transport-Keys sowie dessen Verwendung zur Verschlüsselung des Netzwerk-Keys am Beispiel von Touchlink Commissioning entnehmen. Zwei 32-Bit Identifier-Felder des Frames werden zu einem 128-Bit Wert verkettet. Dabei folgt diese Verkettung folgendem Schema [22]:

```
transaction identifier || transaction identifier || response
identifier || response identifier
```

Dieser 128-Bit Wert wird danach mit dem Master-Key, welchen sehr häufig der Default Trust Center Link Key darstellt, verschlüsselt. Daraus entsteht der Transport-Key, welcher in weiterer Folge verwendet wird, um den Netzwerk-Key zu verschlüsseln. Auf der Gegenseite kann das Zielgerät ebenso denselben Transport-Key ableiten und somit den empfangenen Netzwerk-Key entschlüsseln. [12]

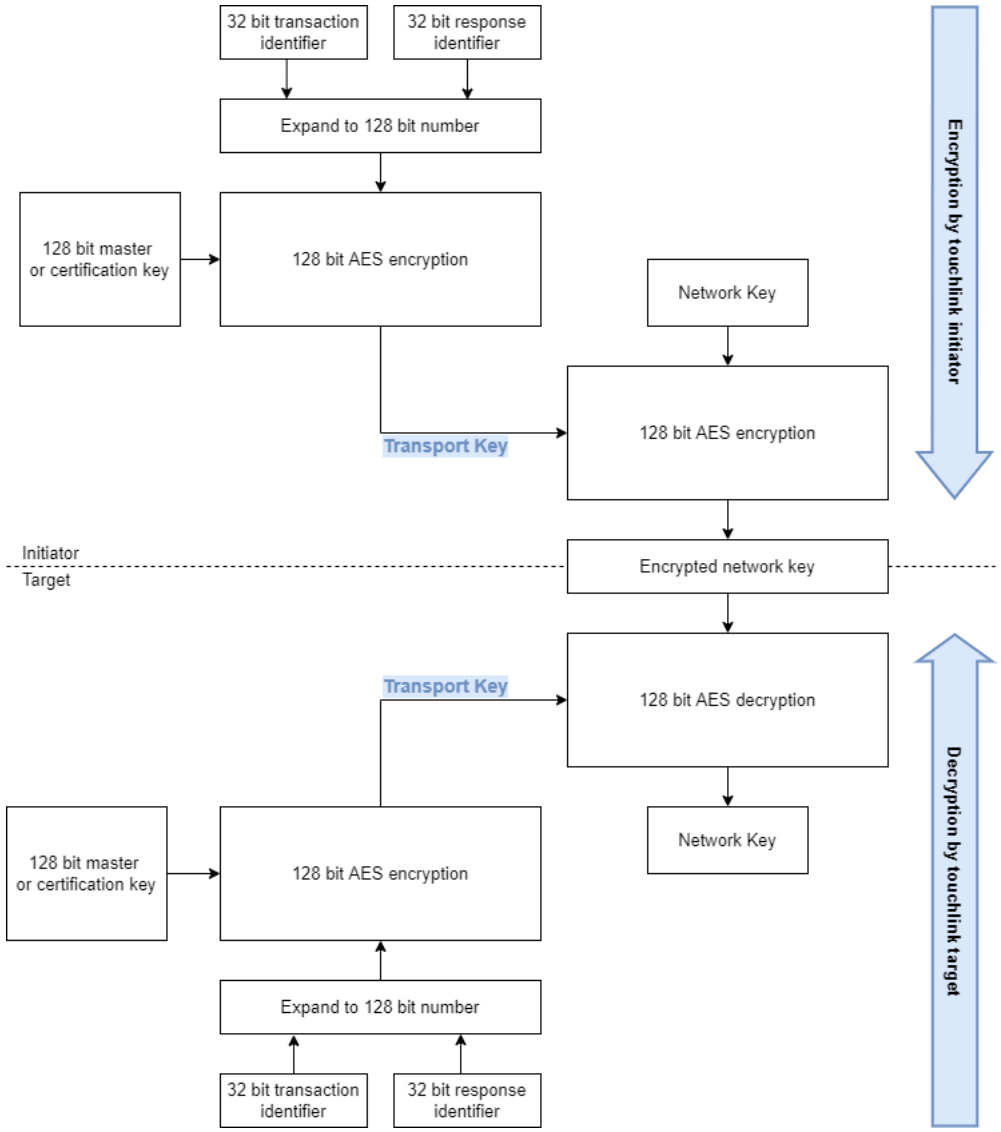

Abbildung 3.4: Transport-Key [22]

3.6.2 Install Codes

Neben dem vorkonfigurierten Link-Key gibt es seit ZigBee 3.0 mit Install Codes eine sichere Alternative. Hierbei wird bereits vom Hersteller ein zufälliger Code in das Gerät integriert, von welchem der ZigBee-Stack dann mittels Matyas-Meyer-Oseas (MMO) Hash Funktion den Link-Key ableiten kann. Dieser Install Code wird beim Hinzufügen des Nodes zum Netzwerk vom User in das Trust Center eingetragen, welches wiederum mittels MMO den gleichen Link-Key ableiten kann. Dieser Link-Key dient in weiterer Folge zur verschlüsselten Übertragung des Netzwerk-Keys. Install Codes können eine Länge von 8, 10, 14 oder 18 Bytes aufweisen. Die letzten beiden Bytes dienen hierbei als CRC. Es wird den Herstellern empfohlen komplett randomisierte Codes mit einer Länge von 18 Byte zu generieren, wel-

che keinen Bezug zu IEEE/MAC Adresse oder sonstigen Produkteigenschaften aufweisen. Somit soll Reverse Engineering und in weiterer Folge das Herausfinden von Ableitungsfunktionen zur Generierung der Install Codes vermieden werden. In Abbildung 3.5 wird diese Variante zum Netzwerkbeitritt grafisch dargestellt. Dieser kann entnommen werden, dass der Sicherheitslevel der Kommunikation als sehr hoch einzustufen ist. Angreifer*innen kennen, im Gegensatz zum vorkonfigurierten Link-Key, hierbei den abgeleiteten Schlüssel nicht. Somit können diese die initiale Übertragung des Netzwerk-Keys nicht entschlüsseln, wodurch auch die restliche Kommunikation als sicher gilt.

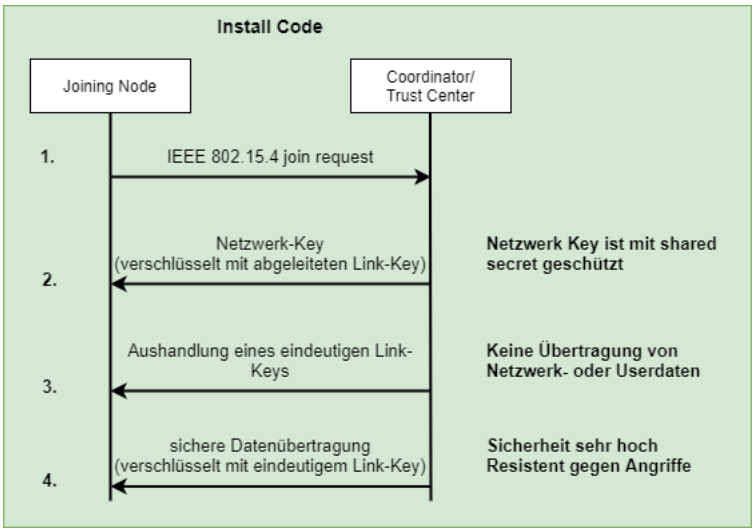

Abbildung 3.5: Install Code [19]

3.6.3 Certificate-based Key Establishment (CBKE)

Es gibt auch ein zertifikatsbasiertes Verfahren zur Erstellung beziehungsweise dem Austausch der privaten Schlüssel, welches beispielsweise von Smart Energy verwendet wird. Dabei müssen das Trust Center sowie alle Teilnehmer ein gültiges Zertifikat hinterlegt haben, welches von einer vertrauenswürdigen Certificate Authority (CA) ausgestellt wurde. Ebenso wird der Public Key der ausstellenden CA, auch CA Root Key genannt, auf den Geräten gespeichert. Auf Grund des Prinzips der Public Key Infrastructure (PKI) ist es durch den Public Key des Zertifikats möglich, ein Gerät eindeutig zu identifizieren und in weiterer Folge eine verschlüsselte Verbindung aufzubauen. Für den Austausch der Schlüssel wird sich einer elliptischen Kurve bedient, wobei das Verfahren ”Elliptic Curve Menezes-Qu-Vanstone (ECMQV)” verwendet wird. Es gibt verschiedene Certificate Authorities von denen die Zertifikate bezogen werden können. Ein Problem hierbei ist, dass auf Grund der sehr geringen Speicherkapazität auf den ZigBee-Geräten nicht alle Root-CA’s als vertrauenswürdig hinterlegt werden können. So wird meist nur einer CA vertraut und man muss bei der Verwendung mehrerer Geräte darauf achten, dass sämtliche Zertifika-

te von derselben CA ausgestellt wurden damit sich die Teilnehmer gegenseitig vertrauen. [19]

Folgend wird die Funktionsweise von CBKE detailliert erklärt, wobei im ersten Schritt Zufallszahlen erzeugt und zusammen mit dem Public Key vom Zertifikat an den anderen Teilnehmer übermittelt werden. Im nächsten Schritt wird ein Key Bitstream erzeugt. Dieser Key Bitstream wird anschließend von den beiden Teilnehmern dazu verwendet um zum einen denselben privaten Schlüssel und zum anderen mittels Hash-Funktion einen MAC (Message Authentication Code) zu erzeugen. Nun senden die Teilnehmer den MAC an den gegenüber und vergleichen diesen mit dem selbst errechneten MAC. Sind diese identisch so haben beide denselben privaten Schlüssel und können die Kommunikation ab sofort mit dem neuen Link-Key verschlüsseln. Dieser Prozess wird in Abbildung 3.6 grafisch veranschaulicht. [24]

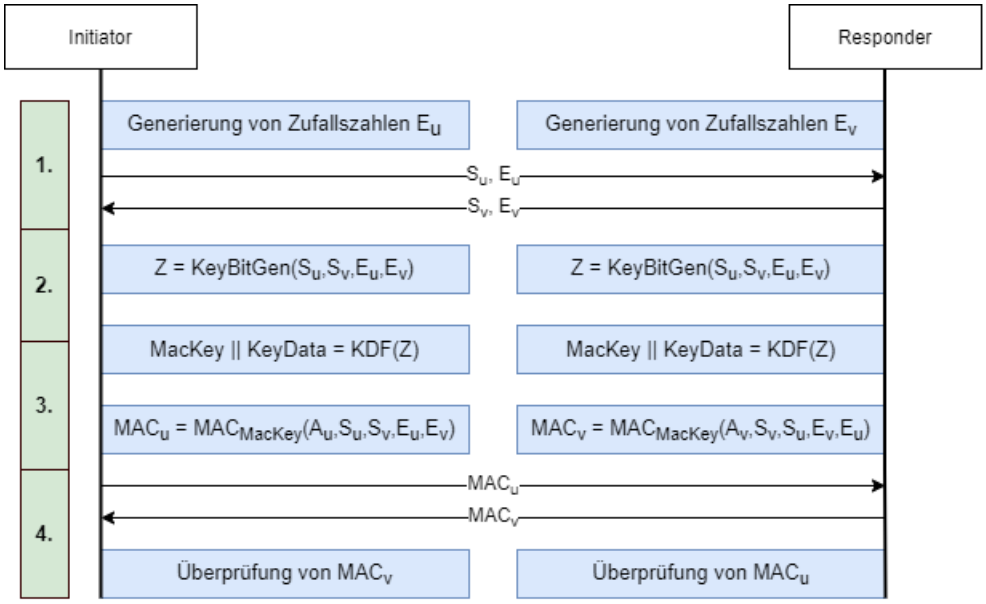

Abbildung 3.6: Funktionsweise CBKE [24]

Folgend werden die einzelnen Prozessschritte von Abbildung 3.6 detailliert beschrieben [24]:

1. E_u und E_v repräsentieren die Zufallszahlen, welche im Detail die Public Keys von zufällig generierten Key-Pairs sind. S_u und S_v sind eine Kombination aus der 64-bit langen Adresse des Gerätes sowie von dessen Public Key.

2. Die Funktion KeyBitGen wird verwendet um den Key Bitstream zu generieren. Da dieser Funktion von beiden Teilnehmern dieselben Parameter übergeben werden, können diese denselben privaten Schlüssel erzeugen.

3. Der KDF (KeyDerivation Function) übergeben beide Teilnehmer unabhängig voneinander den errech-

neten Schlüssel. In diesem Schritt wird ein MAC Key (MacKey) erzeugt, welcher im nächsten Schritt verwendet wird um mittels Keyed-Hash Funktion einen MAC (Message Authentication Code) zu generieren. Bei dieser Funktion wird der MAC mit dem privaten Schlüssel sowie der Matyas-Meyer-Oseas (MMO) Hash Funktion erzeugt. Weiters wird ein Key (KeyData) erstellt, welcher den Link-Key darstellt. Hierbei fließt auch eine zusätzliche Komponente "A" mit ein. Diese ist für den Initiator das Oktekt 02_{16} und für den Responder 03_{16}.

4. Abschließend vergleichen die beiden Teilnehmer die selbst errechnete MAC mit jener, die vom anderen Teilnehmer empfangen wurde. Stimmen diese überein so haben beide denselben Link-Key errechnet.

3.6.4 Sicherheitsaspekte beim Schlüsselmanagement

Forward Secrecy bedeutet, dass wenn ein ZigBee-Gerät das Netzwerk verlässt, dieses keinen Zugriff mehr auf die künftige Kommunikation im Netzwerk haben muss. Hierbei spielt es keine Rolle, ob das Gerät das Netzwerk freiwillig verlässt oder mögliche Angreifer*innen dafür gesorgt haben. Sobald ein ZigBee-Gerät nicht mehr Teil des Netzwerks ist benötigt es auch die dafür gespeicherten Link-Keys sowie den Netzwerk-Key nicht mehr. Leider sind im ZigBee Standard keine Prozesse definiert, die beim Verlassen eines Netzwerks vom Gerät durchgeführt werden müssen. Dies führt zu dem Problem, dass Angreifer*innen ein Gerät infiltrieren und auf das darauf gespeicherte Schlüsselmaterial zugreifen könnten. Mit diesem wäre es in weiterer Folge möglich den gesamten Netzwerktraffic zu entschlüsseln. Weiters könnten diese versuchen das Routing zu sabotieren um die Funktionalität des ZigBee-Netzwerks einzuschränken. [12, 31]

Unter dem Begriff Backward Secrecy versteht man die Geheimhaltung von vergangener Kommunikation beim Zutritt eines neuen Teilnehmers zum Netzwerk. Ein Gerät, welches dem Netzwerk beitritt, soll also keinen Zugriff auf die vergangene Kommunikation haben. Aus diesem Grund sollte jedes Mal wenn ein Teilnehmer dem Netzwerk beitritt der Netzwerk-Key erneuert werden. [12, 31]

Wie in Unterabschnitt 3.6.3 beschrieben gibt es die Möglichkeit zur Authentifizierung und Schlüsselverteilung mittels Zertifikaten. Dabei ist im Key Establishment Cluster festgelegt, dass praktisch jeder bis hin zum/zur Endbenutzer*in Zertifikate ausstellen kann. Da auf den Geräten nur sehr wenig Speicherkapazität vorhanden ist ergibt sich daraus das Problem, dass nicht alle möglichen Public Keys der ausstellenden CA's hinterlegt werden können. Um aber ein Gerät erfolgreich authentifizieren zu können muss auf beiden Teilnehmern der Public Key des Ausstellers vom Zertifikat des jeweilig anderen Geräts installiert sein. [12]

3.7 Sicherheit der Layer

Die Sicherheit des MAC-Layers basiert auf der Sicherheit der IEEE 802.15.4 Spezifikation sowie CCM (Counter mit CBC-MAC). Der CCM besteht aus einem Counter sowie dem CBC-MAC Verfahren, wodurch Vertraulichkeit und Integrität sichergestellt werden kann. Eine Aufgabe des MAC-Layers ist es seine Frames zu schützen, wobei das zu verwendende Sicherheitslevel und die Keys von den darüberliegenden Layern vorgegeben werden. Dabei werden die aktiven Link-Keys sowie der aktive Netzwerk-Key verwendet. [26]

Die Sicherheit auf Netzwerk-Layer Ebene wird durch AES Verschlüsselung mit einer Schlüssellänge von 128-Bit sowie der Verwendung eines CBC-MAC gewährleistet. Der hierfür verwendete Netzwerk-Key, den sich alle Netzwerkteilnehmer teilen, wird wie in Abschnitt 3.6 erklärt nach erfolgreicher Authentifizierung des ZigBee-Gerätes beim Zutritt zum Netzwerk vom Trust Center übermittelt. Die Verschlüsselung dieser initialen Übertragung des Keys kann mit einem vorkonfigurierten Link-Key, einem Install Code oder einem zertifikatsbasierten Verfahren gesichert werden. Die Geräte verwenden den Netzwerk-Key in erster Linie zur Ver- und Entschlüsselung von protokollspezifischen Daten auf Netzwerk-Layer Ebene. Des Weiteren wird der Netzwerk-Key in manchen Applikationen auch für die Verschlüsselung von Benutzerdaten verwendet. [11]

Innerhalb des Application-Layers (APL) befindet sich der Application Support Sublayer (APS). Dieser Sublayer kümmert sich um sämtliche sicherheitsspezifische Themen, die im APL angesiedelt sind. Im Application-Layer besteht wie bereits erwähnt die Möglichkeit, dass zwei Geräte eine verschlüsselte End-to-End Verbindung zueinander aufbauen. Dabei teilen sich die beiden kommunizierenden Geräte einen privaten Schlüssel, welcher nur diesen beiden bekannt ist. Dieser Key, mit dem die Daten auf Applikationslayer Ebene verschlüsselt werden, wird auch Application-Key oder Link-Key genannt. Das Trust Center kümmert sich um die Generierung und Verteilung des Schlüssels an die beiden Teilnehmer. Somit bietet dies eine zusätzliche Sicherheitsschicht zur Verschlüsselung mit dem Netzwerk-Key auf Netzwerk-Layer Ebene. Ein Anwendungsfall für diese zusätzliche Sicherheitsmaßnahme wäre beispielsweise ein Smart Home das Thermostate, Lichtsteuerung, Türschlösser und Garagentoröffner enthält. Dabei ist es sinnvoll die Kommunikation zwischen Türschlössern und Garagentoröffner zusätzlich mit einem Link-Key zu verschlüsseln. Dadurch ist gewährleistet, dass ein/eine Angreifer*in nicht bereits durch Erlangen des Netzwerk-Keys Nachrichten an alle Teilnehmer senden kann. Dazu wäre zusätzlich noch der Link-Key, den sich Türschloss und Garagentoröffner teilen, notwendig. Des Weiteren wird beim initialen Zutritt zum Netzwerk die Übertragung des Netzwerk-Keys vom Trust Center zum beitretenden

Gerät mittels Link-Key gesichert. Wenn für das Netzwerk das Attribut ”Link Key Security” aktiviert ist, dann wird die künftige Kommunikation mit dem Trust Center auch mit einem eindeutigen Link-Key verschlüsselt. Weiters verwendet das Gerät diesen Link-Key im Falle eines Rejoins um dem Netzwerk erneut beizutreten. [11, 26]

3.8 Over-the-air (OTA) Updates

ZigBee bietet den Herstellern mit over-the-air Updates die Möglichkeit die Sicherheit von ZigBee-Netzwerken nachträglich zu erhöhen oder hochzuhalten. Somit können Fehlfunktionen in Produkten behoben und neue Features oder Sicherheitspatches eingespielt werden. Auf der anderen Seite birgt das Einspielen von Updates auch ein mögliches Risiko, falls sich die Hersteller nicht an die sicherheitsspezifischen Vorgaben halten. ZigBee bietet hierbei ein mehrstufiges Sicherheitskonzept damit sichergestellt werden kann, dass die Code-Images nicht manipuliert werden können. Dabei werden alle OTA-Images bei der Herstellung signiert und mit einem eigenen Key verschlüsselt. Des Weiteren werden sämtliche Übertragungen von OTA-Images verschlüsselt. Um Standard Reverse Engineering Methoden zu vermeiden kann das OTA-Image auf einem On-Chip Speicher gespeichert werden, auf welchem die Debugging Funktion deaktiviert ist. Wenn nun ein Teilnehmer ein neues Image via Update erhält, wird dieses vom Bootloader entschlüsselt, die Signatur validiert und danach das Update eingespielt. Eine weitere Aufgabe des Bootloader ist die Validierung der Images beim Start des Geräts, wodurch dieser im Falle eines fehlerhaften Updates sofort auf das vorherige Image zurückwechseln kann. [26]

3.9 Möglichkeiten für Rejoin zum Netzwerk

Beim Zutritt eines neuen Teilnehmers zum Netzwerk wird, wie in Abschnitt 3.6 detailliert beschrieben, der Netzwerk-Key vom Trust Center an das neue Gerät übertragen. Diese initiale Übertragung des Netzwerkschlüssels wird mittels Link-Key verschlüsselt. Falls nun ein Gerät die Verbindung zum Netzwerk verliert, kann es diesem mittels Rejoin erneut beitreten. Im ersten Schritt versucht es der Teilnehmer mit einem sogenannten ”secure rejoin”, wobei die Verbindung mit dem aktiven Netzwerk-Key verschlüsselt wird. Diese Möglichkeit besteht bei zentralen sowie verteilten ZigBee-Netzwerken. Des Weiteren ist auf Grund der Verschlüsselung auf Netzwerk-Layer Ebene keine weitere Autorisierung erforderlich. Dieser Prozess kann Abbildung 3.7 entnommen werden. [17]

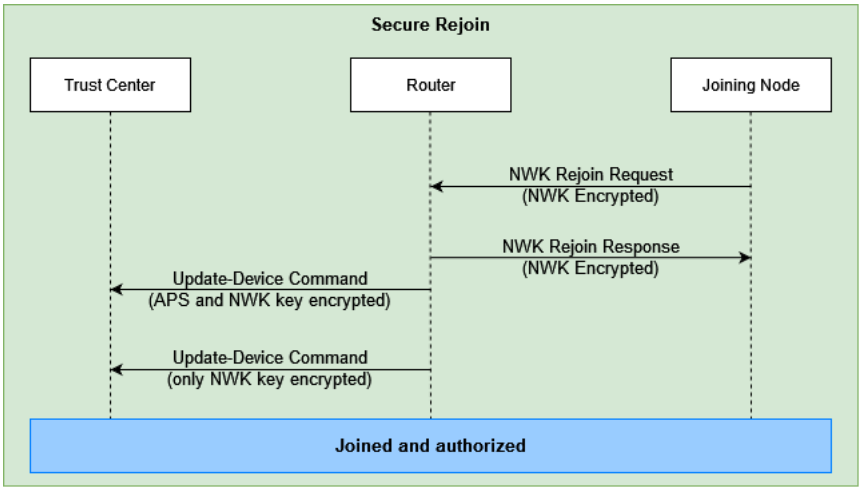

Abbildung 3.7: Secure Rejoin [17]

Wurde der Netzwerkschlüssel während der Abwesenheit des Geräts ausgetauscht so schlägt dieser Versuch fehl und es wird im nächsten Schritt ein ”Insecure Rejoin”, welcher auch Trust Center Rejoin genannt wird, durchgeführt. Hierbei fordert der Teilnehmer den aktuellen Netzwerk-Key an, wobei der Trust Center Link Key zur Verschlüsselung auf Application-Layer Ebene verwendet wird. Des Weiteren findet auf Netzwerk-Layer Ebene keine Verschlüsselung zwischen dem beitretenden Gerät und seinem Parent statt. Sollte auch diese Variante des Rejoins fehlschlagen, so startet das Gerät den Standardprozess um dem Netzwerk als neues Gerät beizutreten. Das Problem in diesem Szenario ist der Insecure Rejoin. Im Falle vom Home Automation Profil wird immer derselbe Link-Key verwendet um die Übertragung zu verschlüsseln. Wird nun diese initiale Übermittlung durch Angreifer*innen abgehört können diese den Netzwerk-Key entschlüsseln und das Netzwerk kompromittieren. Der detaillierte Ablauf des Angriffs sowie mögliche Gegenmaßnahmen können Abschnitt 4.1 entnommen werden. [19, 12, 16]

3.10 Touchlink Commissioning (ZLL)

Das Feature ”Touchlink Commissioning” ermöglicht es, dass ein Gerät einem Netzwerk ohne Userinteraktion beitritt. Die Sicherheit eines solchen ZigBee-Netzwerkes besteht also in der Geheimhaltung des Netzwerk-Keys, welcher beim Beitritt zum Netzwerk an das beitretende Gerät übermittelt wird. Diese initiale Übertragung wird mittels, auf sämtlichen Geräten vorkonfigurierten, Light Link Master-Key verschlüsselt. Das Problem hierbei ist, dass dieser Default Key öffentlich bekannt ist und somit die initiale Verbindung entschlüsselt werden kann. Dadurch ist es möglich den Netzwerkschlüssel zu extrahieren und in weiterer Folge die gesamte Kommunikation auf Netzwerk-Layer Ebene zu entschlüsseln. Ein weiterer Nachteil ist, dass die Änderung dieses Default Keys in zertifizierten ZigBee-Geräten nicht mög-

lich ist. In Abbildung 3.8 wird ein Überblick über die Sicherheit von Touchlink gegeben. Dieser kann entnommen werden, dass beide Geräte den Default Touchlink-Key hinterlegt haben müssen, damit sie den Netzwerk-Key ver- beziehungsweise entschlüsseln können. Die Übertragung des Netzwerk-Keys wird also lediglich mit dem Default-Key verschlüsselt. Dadurch ist es möglich, dass potenzielle Angreifer*innen diese Nachrichten abfangen und den Netzwerk-Key entschlüsseln. Nachdem das Zielgerät im Besitz des Netzwerk-Keys ist wird die Kommunikation auf Netzwerk-Layer mit diesem verschlüsselt. Beim ”ID exchange” werden verschiedene Parameter wie beispielsweise der verwendete Verschlüsselungsalgorithmus ausgehandelt. In diesem Schritt übermittelt das Zielgerät dem Initiator per Scan Response die unterstützten Algorithmen. Da sämtliche dieser Algorithmen den hinterlegten Default Key verwenden erhöht sich durch die Vielfalt an Algorithmen die Sicherheit nicht. Dies dient lediglich zur Verbesserung der Interoperabilität. [22, 21]

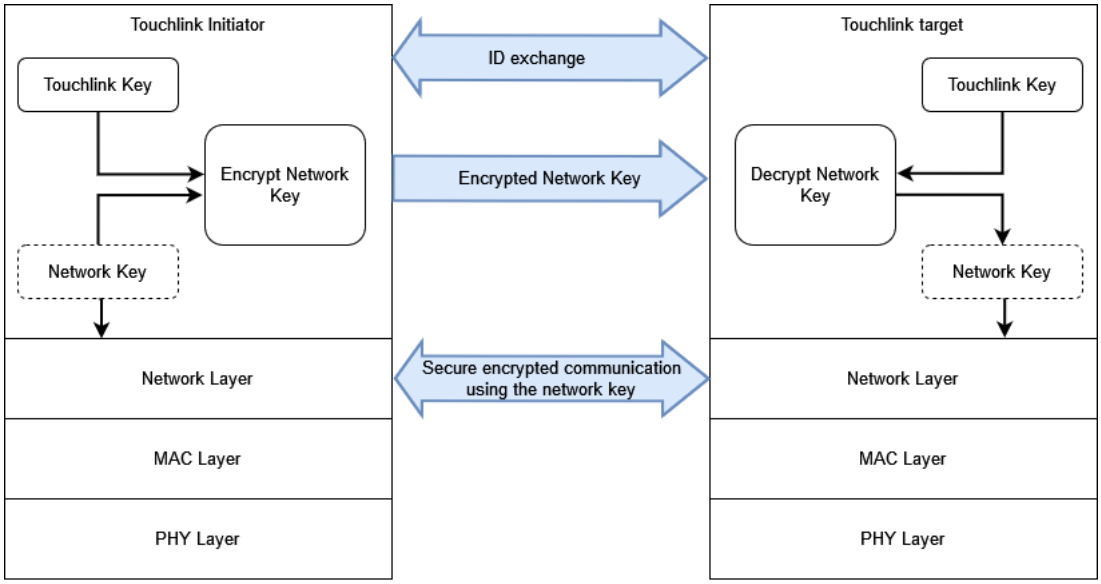

Abbildung 3.8: Sicherheit von Touchlink Commissioning [23]

4 Angriffe auf ZigBee-Netzwerke

Es gibt eine Vielzahl von Arbeiten mit theoretischen Angriffen auf ZigBee-Netzwerke. Bei der Evaluierung der praktisch möglichen Angriffe wurde festgestellt, dass sich diese im Wesentlichen auf vier grundlegende Typen beschränken. Im folgenden Kapitel werden diese detailliert beschrieben und auch auf praktische Durchführungen sowie Angriffe aus der Literatur eingegangen. Folgend werden die vier grundlegenden Angriffe auf ZigBee-Netzwerke aufgelistet:

- Insecure Rejoin
- Übernahme von Geräten
- Replay
- Netzwerk-Key Sniffing

4.1 Insecure Rejoin

Beim Insecure Rejoin handelt es sich, wie in Abschnitt 3.9 beschrieben, um eine Möglichkeit mit der ein ZigBee-Gerät einem Netzwerk erneut beitreten kann. Dieser wird verwendet, sollte sich während dessen Abwesenheit der Netzwerk-Key geändert haben. Dabei werden im Unterschied zum ”Secure Rejoin” der Rejoin Request sowie der Rejoin Response unverschlüsselt zwischen Parent Gerät und dem Teilnehmer übertragen. Diesen Umstand können Angreifer*innen ausnutzen um die Übertragung des Netzwerk-Keys abzufangen und somit das Netzwerk zu kompromittieren. [16]
T. Zillner veranschaulicht in seinem Paper ”ZIGBEE EXPLOITED The good, the bad and the ugly” die Möglichkeit den Netzwerk-Key im Zuge eines Insecure Rejoins mitzulesen. Dabei stören Angreifer*innen die Kommunikation zwischen Trust Center und dem ZigBee-Gerät, bis dieses aus dem Netzwerk entfernt wird. Im Zuge vom darauffolgenden Insecure Rejoin sendet das beitretende Gerät einen unverschlüsselten ”NWK Rejoin Request” an seinen Parent, welcher mit einem ”NWK Rejoin Response” darauf antwortet. Im nächsten Schritt wird das beitretende Gerät, derzeit noch unautorisiert, zum ZigBee-Netzwerk hinzugefügt. Das Parent Gerät schickt ein ”Update-Device Command” an das Trust Center,

woraufhin dieses über den Zutritt des Gerätes entscheidet. Falls das Gerät in der Network Information Base des Trust Centers gespeichert ist, wird der Netzwerk-Key mittels APS und NWK Verschlüsselung an das Parent Gerät gesendet. Dieses leitet den aktiven Netzwerk-Key via ”Transport Key Kommando” an das Endgerät mittels APS-Layer Verschlüsselung weiter. Für diese Verschlüsselung wird, wie beim initialen Netzwerkbeitritt, der Default Trust Center Link Key verwendet. Dadurch ist es einem/einer Angreifer*in möglich den übertragenen Netzwerk-Key zu entschlüsseln. Als Beispiel zeigt T. Zillner wie unter Ausnutzung dieser Schwachstelle ein/eine Angreifer*in ein smartes Türschloss entsperren kann, ohne dass dies die Benutzer*innen bemerken. Dabei wurde im ersten Schritt der Netzwerk-Key extrahiert und anschließend Befehle an das Türschloss gesendet, um dieses zu entsperren. In Abbildung 4.1 wird der Prozess von einem Insecure Rejoin grafisch dargestellt. [21, 17]

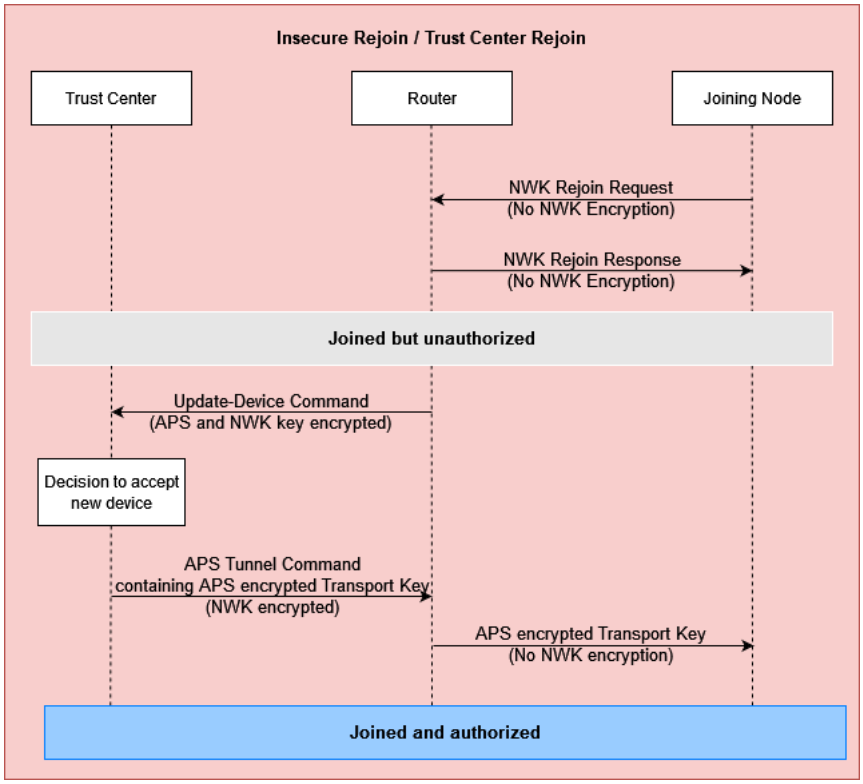

Abbildung 4.1: Insecure Rejoin [17]

In Abbildung 4.2 wird der praktische Ablauf eines Insecure Rejoins in Wireshark gezeigt. Dieser kann entnommen werden, dass zuerst ein unverschlüsselter Rejoin Request sowie der zugehörige Response übertragen werden. Des Weiteren ist die darauf folgende Übertragung des Netzwerk-Keys, welche mit dem Default Trust Center Link Key verschlüsselt ist, ersichtlich.

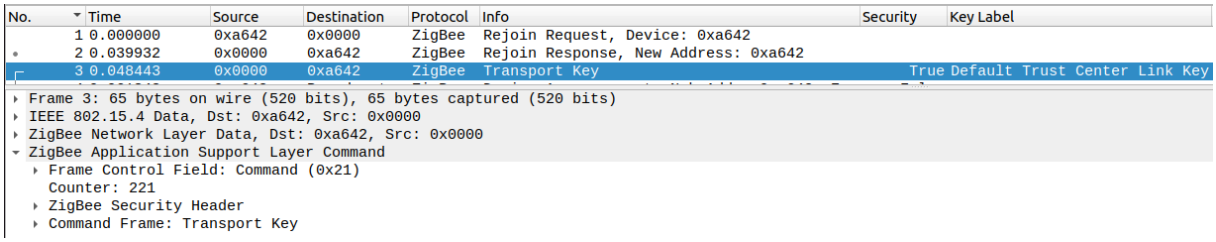

Abbildung 4.2: Beispiel Insecure Rejoin

4.1.1 Gegenmaßnahmen

Prinzipiell ist auch ZigBee 3.0 aus Gründen der Abwärtskompatibilität noch anfällig für diesen Angriff. Dabei können Hersteller selbst entscheiden, ob sie dieses Feature implementieren. Beim SmartThings Hub können Insecure Rejoins beispielsweise per App unterbunden werden. Nichtsdestotrotz wurden im Standard einige Verbesserungen vorgenommen, mit denen der Angriff vermieden werden kann. Um dies zu gewährleisten, sollte unter ZigBee 3.0 der Mechanismus zur Schlüsselgenerierung beim Netzwerkbeitritt verwendet werden. Hierbei wird, nachdem ein Gerät dem ZigBee-Netzwerk beigetreten ist, vom Trust Center ein neuer randomisierter Link-Key übermittelt, welcher für künftige Rejoins des Gerätes verwendet wird. Da die Angreifer*innen diesen Link-Key nicht kennen, können diese auch bei Rejoins den Netzwerk-Key nicht mehr entschlüsseln. Eine weitere Schutzmaßnahme wäre auf dem Trust Center das automatisierte Rejoinen via Trusted Link-Key per Policy zu unterbinden. Dadurch kann ein Gerät dem Netzwerk nicht mehr automatisch mit einem vorkonfigurierten Link-Key beitreten, sondern das Gerät verbindet sich wie beim initialen Beitritt mit dem Netzwerk wodurch eine Interaktion mit dem/der Benutzer*in notwendig wird. [19, 16] Eine weitere Maßnahme zur Verringerung des Risikos wäre die Verwendung des im Zuge dieser Arbeit entwickelten Network Intrusion Detection Systems (NIDS) um einen Insecure Rejoin zu erkennen. Diese Lösung bietet den Vorteil, dass die mögliche Kompromittierung des Systems sofort erkannt wird und sich die Angreifer*innen nicht unbemerkt im Netzwerk ausbreiten können. Die Funktionsweise zur Erkennung des Angriffs wird in Unterabschnitt 5.5.1 beschrieben.

4.2 Übernahme von Geräten

Wie bereits erwähnt gibt es im ZigBee Light Link (ZLL) Profil das Feature ”Touchlink Commissioning” wodurch Geräte ohne User-Interaktion zu einem bestehenden Netzwerk hinzugefügt werden können. Dies kann von Angreifer*innen ausgenutzt werden, um ZigBee-Geräte zu übernehmen und dem/der Besitzer*in den Zugriff darauf zu entziehen.

Bei diesem Angriff erstellen die Angreifer*innen selbst ein Netzwerk und übernehmen die Rolle des

Initiators. Dabei senden diese einen ”Reset to Factory” Befehl an alle erreichbaren Netzwerkteilnehmer. Sobald ein Gerät, wie beispielsweise eine Glühbirne, dieses Kommando empfängt verlässt dieses im nächsten Schritt sein derzeitiges Netzwerk und sucht nach einem verfügbaren Netzwerk in seiner Umgebung, um diesem beizutreten. Die Angreifer*innen senden nun Scan Requests an alle Teilnehmer, sodass sich diese zu ihrem Netzwerk hinzufügen. Somit können diese alle Geräte in ihrem Netzwerk beliebig steuern, wohingegen die Benutzer*innen keine Kontrolle mehr über diese haben. Der Umstand, dass weder physischer Zugriff zum Gerät notwendig noch der aktuelle Netzwerk-Key bekannt sein muss, machen dies zu einem schwerwiegenden Sicherheitsproblem. Somit können über Distanzen von mehreren Metern Geräte aus dem Netzwerk der Besitzer*innen gestohlen und kontrolliert werden. [19, 16]
In Abbildung 4.3 wird der Ablauf dieses Angriffs im Detail dargestellt. Dabei sendet der/die Angreifer*in in erster Instanz einen ”Scan Request” um das Zielgerät dazu zu bringen die verfügbaren Channel zu scannen. Im nächsten Schritt wird der ”Reset to Factory” Befehl in Form eines Inter-PAN Kommandos übertragen. Dies muss lediglich die Source, Destination Adresse und den Wert 0x07 als Command ID gesetzt haben damit das Zielgerät sein aktuelles Netzwerk verlässt. Danach wird dieses mittels ”Network Join End Device Request” dazu aufgefordert sich mit dem Netzwerk der Angreifer*innen zu verbinden. Die ”.ind” Pakete beschreiben den Datentransfer der NSDU vom NWK-Layer zum lokalen APS Sub-Layer. Die ”.conf” Pakete dienen als Bestätigung, welchen den Status der Anforderung beinhalten. Die ”.req” Pakete beschreiben interne Requests zwischen NWK- und APS-Layer.

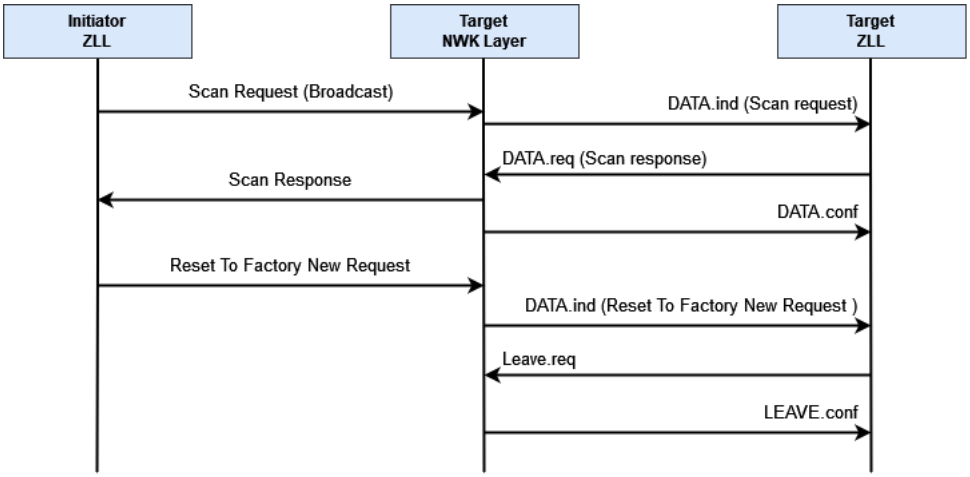

Abbildung 4.3: ZLL Reset to Factory [16]

Alternativ zum ”Factory to Reset” Befehl gibt es auch noch weitere Möglichkeiten, um ein ZigBee-Gerät auf Werkseinstellungen zurück zu setzen. So kann ein ZLL Gerät auch durch das Trennen der Energiezufuhr auf Werkseinstellungen zurückgesetzt werden. Des Weiteren gibt es herstellerspezifische Unterschiede mit welchen Tastenkombinationen dies ebenso möglich ist. Im Falle von Glühbirnen gibt

es auch oft ”Energiezyklen” wobei sich ein Gerät nach mehreren Zustandsänderungen innerhalb eines definierten Zeitabstandes zurücksetzt. [16]

4.2.1 Gegenmaßnahmen

Sobald sich ein Gerät im Netzwerk befindet sollte Touchlink Commissioning auf diesem deaktiviert werden, um diese Gefahr zu vermeiden. Ein weiterer Schritt wäre das Gerät so zu konfigurieren, dass es keine Inter-PAN Touchlink Nachrichten mehr verarbeitet. [19] Ebenso wäre die Verwendung vom Network Intrusion Detection System (NIDS) von Vorteil. Dieses kann die Übernahme der Geräte zwar nicht verhindern, aber sofort darüber informieren, wenn ein Gerät das eigene Netzwerk verlässt. Somit kann sofort darauf reagiert werden bevor die Angreifer*innen dies ausnutzen können. Der Ablauf des Angriffs bei Verwendung des NIDS kann Unterabschnitt 5.5.3 entnommen werden.

4.3 Replay

Unter einem Replay Angriff versteht man das erneute Senden von zuvor aufgezeichneten Paketen. Dabei wird versucht das ZigBee-Gerät dazu zu bringen eine bestimme Aktion erneut auszuführen. Bei diesem Angriff müssen weder Netzwerk-Key noch Link-Keys bekannt sein, wodurch die Angreifer*innen nicht teil des Netzwerks sein müssen, um diesen durchzuführen. Im Bereich der smarten Türschlösser kann beispielsweise eine erneute Übertragung des ”Tür öffnen” Befehls schwere Auswirkungen für die Beteiligten haben. Wie bereits in Unterabschnitt 3.2.5 beschrieben werden in ZigBee Frame Counter verwendet, um Replay-Angriffe zu unterbinden. Diese Frame Counter werden jedoch bei ZigBee 2012 Geräten beim Neustart beziehungsweise beim Reset eines Geräts zurückgesetzt. Danach kann ein/eine Angreifer*in zuvor aufgezeichnete Pakete erneut übermitteln, wobei diese auf Grund des höheren Frame Counters akzeptiert und ausgeführt werden. Um ZigBee-Geräte zurückzusetzen gibt es verschiedene Möglichkeiten, die sich anhand deren physischen Zugriffsmöglichkeiten unterscheiden. Sofern man physikalischen Zugriff zum Gerät hat, verfügen diese oft über einen entsprechenden Reset Button, wobei es herstellerspezifische Unterschiede gibt. Ist der Zugriff lediglich über eine Funkverbindung möglich so kann beispielsweise die Batterie von Geräten durch eine Flut an Anfragen entleert werden. Zwei Ausprägungen dieses Angriffs werden in Unterabschnitt 4.5.4 genauer beschrieben.
Seit ZigBee 3.0 gibt es seitens des Standards die Vorgabe, dass der Frame Counter nicht zurückgesetzt werden darf. Lediglich bei der Ausstellung eines neuen Netzwerk-Keys durch das Trust Center darf der Frame Counter zurückgesetzt werden. Nach einer Rotation des Netzwerk-Keys sind aber die aufgezeichneten Pakete nicht mehr gültig, da diese mit dem mittlerweile ungültigen Schlüssel verschlüsselt wurden.

Durch diese Maßnahme wird der Schutz gegen Replay Angriffe erhöht. [16, 17]
Ein Replay Angriff kann in drei Prozessschritte aufgeteilt werden [16]:

1. Initial wird die ZigBee-Kommunikation von einem/einer Angreifer*in aufgezeichnet. Hierbei handelt es sich beispielsweise um den ”Tür öffnen” Befehl in verschlüsselter Form.

2. Im nächsten Schritt bringt der/die Angreifer*in das Gerät dazu sich neu zu starten oder auf Werkseinstellungen zurückzusetzen, sodass der Frame Counter neu initialisiert wird.

3. Der/Die Angreifer*in übermittelt nun die zuvor aufgezeichneten, immer noch verschlüsselten, ZigBee-Pakete an das Zielgerät.

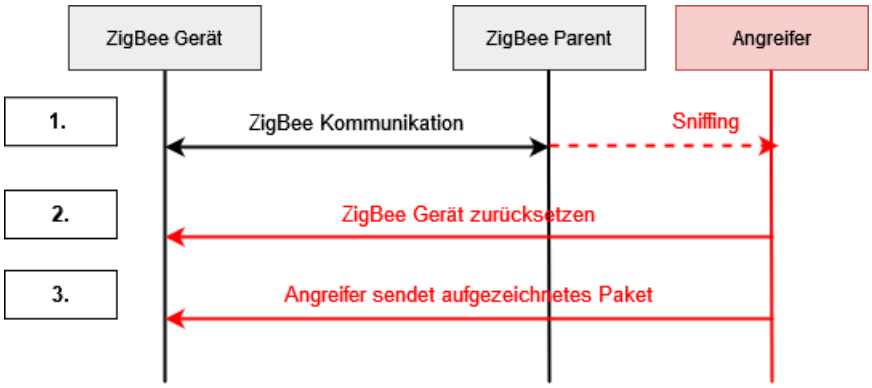

Abbildung 4.4: Replay [16]

4.3.1 Gegenmaßnahmen

Um Replay Angriffe bestmöglich zu vermeiden ist die Verwendung von ZigBee 3.0 spezifizierten Geräten zu empfehlen. Für bestehende Installationen empfiehlt sich das im Zuge dieser Arbeit entwickelte NIDS zu verwenden um Angriffe erkennen und schnell darauf reagieren zu können. Als Beispiel sei hier wiederum das Öffnen von einem smarten Türschloss genannt. Wartet ein/eine Angreifer*in bis die Besitzer*innen das Eigenheim verlassen um anschließend die Tür mittels Replay zu öffnen würde dies ohne NIDS wohl unbemerkt bleiben. Wird hingegen das entwickelte System eingesetzt so werden die Besitzer*innen sofort darüber informiert, dass in diesem Moment ein Angriff auf das Türschloss stattfindet. Je nach verfügbaren smarten Komponenten könnten die Besitzer*innen auf ihre Überwachungskameras zugreifen, um sich sofort aus der Ferne einen Überblick über die Situation im Eigenheim zu verschaffen und im Ernstfall weitere Maßnahmen zu ergreifen. Das Erkennen eines Replay-Angriffs durch das NIDS folgt in Unterabschnitt 5.5.2.

4.4 Netzwerk-Key Sniffing

Wie bereits mehrfach erwähnt wird beim Beitritt zu einem Netzwerk die initiale Übertragung des Netzwerk-Keys sehr häufig mit dem Default Trust Center Link Key verschlüsselt. Angreifer*innen können somit diese Übertragung entschlüsseln, wenn diese während eines Netzwerkbeitritts den Netzwerkverkehr aufzeichnen. Da dies in der Regel nicht oft vorkommen wird bedienen sich diese dem Faktor Mensch. Im ersten Schritt ermitteln die Angreifer*innen den verwendeten ZigBee-Channel mittels Sniffing. Danach senden diese kontinuierlich Störsignale aus, um die reguläre ZigBee-Kommunikation zu stören. Somit funktionieren beispielsweise das Türschloss oder die Lampen nicht mehr ordnungsgemäß, da diese die gültigen Befehle nicht mehr interpretieren können. Nun ist es sehr wahrscheinlich, dass die Benutzer*innen das Gerät aus dem Netzwerk entfernen und erneut hinzufügen in der Hoffnung, dass dieses danach wieder auf die Befehle reagiert. In diesem Zug wird der Netzwerk-Key an das Gerät übertragen und kann durch die Angreifer*innen aufgezeichnet werden. Diese haben nun Zugriff auf das ZigBee-Netzwerk und können die weitere Kommunikation kompromittieren. Da zu diesem Zeitpunkt auf dem Gerät noch keine Link-Keys vorhanden sind und deren Übertragung mittels Netzwerk-Key verschlüsselt wird haben die Angreifer*innen in weiterer Folge auch Zugriff auf die Link-Keys des Geräts. [21] In Abbildung 4.5 werden die Prozessschritte beim Beitritt zu einem Netzwerk inklusive der problematischen Übertragung des Netzwerkschlüssels dargestellt. Dieses Angriffsszenario beruht auf der Annahme, dass als APS Transport-Key der häufig verwendete Default Trust Center Link Key zur Verschlüsselung verwendet wird.

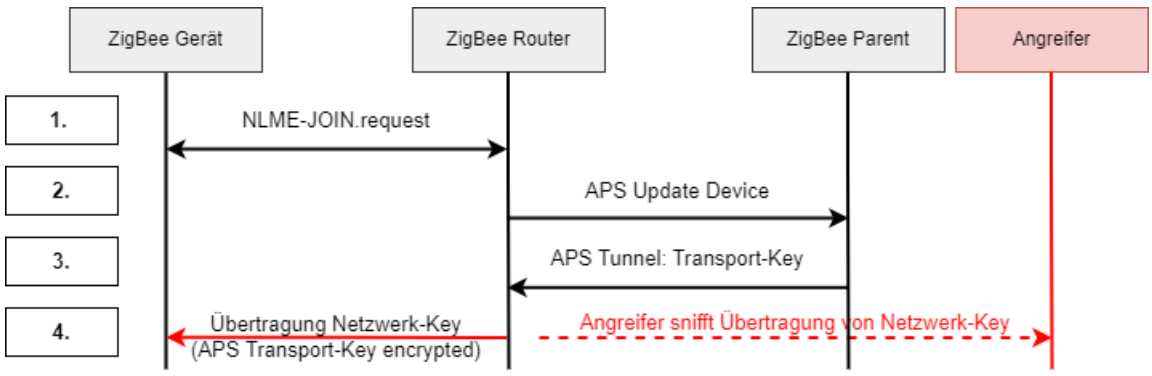

Abbildung 4.5: Netzwerk-Key Sniffing [16]

4.4.1 Gegenmaßnahmen

Um die Kompromittierung des Netzwerk-Keys zu vermeiden sollten wie in Abschnitt 3.6 beschrieben keine vorkonfigurierten Schlüssel verwendet werden. Eine sichere Alternative ist das Verwenden von Install Codes. Leider gibt es in ZigBee 3.0 aus Gründen der Abwärtskompatibilität noch die Möglichkeit

die initiale Übertragung des Netzwerk-Keys mittels vorkonfigurierten Link-Key zu verschlüsseln. Eine weitere Maßnahme wäre wiederum die Verwendung des entwickelten NIDS, wobei der zugehörige Ablauf zur Erkennung in Unterabschnitt 5.5.4 detailliert dargestellt wird.

4.5 Angriffe aus der Literatur

4.5.1 IoT Worm Hack on Philips Hue Light Bulbs

Im November 2016 wurde ein Paper veröffentlicht, in welchem Security Researcher einen Proof-of-Concept eines ZigBee-Wurms entwickelt haben. Dieser verbreitet sich selbständig auf alle erreichbaren Philips Hue Lampen und bringt diese dazu das Licht aus- und einzuschalten. Dabei schleusten die Angreifer*innen mit Hilfe einer Drohne einen Wurm in das ZigBee-Netzwerk ein. Um den Erfolg des Angriffs aufzuzeigen, morsten die Researcher das SOS Signal mit Hilfe der infiltrierten Lampen. Diese nutzten, von Philips in der Firmware hinterlegte, Keys aus um sich im Netzwerk auszubreiten. So ist es möglich Glühbirnen aus bis zu 400 Metern Entfernung zu kompromittieren. Laut den Researchern ist es mit diesem Angriff möglich binnen Minuten eine ganze Smart City, welche Philips Lampen verwendet, zu infiltrieren und fernzusteuern. Weiters wäre es möglich diese Lampen für einen DDOS-Angriff zu missbrauchen. Nachdem diese Sicherheitslücke bekannt wurde veröffentlichte Philips umgehend Firmware Updates mit denen sich die Besitzer*innen von Hue Lampen gegen diesen Angriff schützen können. [32, 26]

4.5.2 Replay Angriff auf Philips Hue

In ihrem Paper [33] haben Mohammad Shafeul Wara und Qiaoyan Yu eine neue Art von Replay Angriff entwickelt, welche auch bei ZigBee 3.0 und der aktuellen Philips Hue Version funktioniert. Replay Angriffe sind sehr leicht zu implementieren und können verheerende Auswirkungen auf die Zielsysteme haben. Bei dem vorgestellten Angriff wird verschlüsselter Traffic aufgezeichnet und erneut gesendet, wobei die geheimen Schlüssel nicht bekannt sein müssen. Anhand zweier Fallbeispiele haben diese die Funktionalität ihres Angriffs mittels Xbee und Philips Hue Geräten veranschaulicht und bewiesen. Mit dem Replay Angriff ist es möglich Philips Hue Lampen ein- und auszuschalten, sofern zwischen dem Aufzeichnen der Nachricht und dem Replay Angriff die Stromversorgung des Geräts getrennt wird, um den Frame Counter zurückzusetzen. Dabei entwickelten sie eine Methode zum Entfernen von aufgezeichneter Noise bei Xbee Kommunikation. Xbee's sind hierbei kleine Funkgeräte, welche als Transceiver arbeiten. Beim Evaluieren verschiedener Gegenmaßnahmen wurde festgestellt, dass das Erhöhen der

Frame Counter die Integritätsprüfung (MIC - Message Integrity Check) von Nachrichten nicht beeinflusst. Ebenso wird erläutert, dass diese Erhöhung des Counters für die Authentifizierung von eingehenden Paketen den neuen Replay Angriff nicht verhindern kann. In ihrem Setup verwenden sie zwei Xbee Devices, wobei eines als Coordinator und das andere als Router fungiert. Des Weiteren verwenden sie zwei API-Mote v4 Beta für das Sniffen sowie das erneute Senden der aufgezeichneten Pakete.

Ablauf des Angriffs

Im ersten Schritt wird der gewünschte Traffic zwischen den XBee ZigBee-Geräten mit dem API-Mote 1 aufgezeichnet und als Capture gespeichert. Im nächsten Schritt wird das aufgezeichnete Paket erneut via API-Mote 1 an das Zielgerät gesendet. Diese Übertragung beinhaltet das Originalpaket inklusive einer gewissen Noise, welche durch die Verwendung von Xbee entsteht. Dieser Replay von API-Mote 1 wird wiederum von API-Mote 2 aufgezeichnet und als Capture gespeichert. Im nächsten Schritt werden die beiden Captures verglichen und die Noise herausgerechnet. Diese wird anschließend entfernt, wodurch der Message Integrity Check am Zielgerät erfolgreich sein wird. Für diesen Schritt wird der Paket Dump in das Daintree SNA Format konvertiert, da man dieses einfach per Texteditor bearbeiten kann, um so die Noise zu entfernen. Abschließend kann das überarbeitete Paket wieder an das Zielgerät gesendet werden, woraufhin dieses den Befehl erneut ausführen wird sofern in der Zwischenzeit der Frame Counter mittels Stromentzug zurückgesetzt wurde. Im nächsten Schritt wurden die Xbee Geräte gegen Philips Hue getauscht und der Versuch erneut durchgeführt. Dabei wurde festgestellt, dass keine Noise mehr in dem Capture enthalten ist. Da bei Philips Hue Geräten neben dem Message Integrity Code ebenso ein Frame Counter zur Überprüfung der Gültigkeit von Nachrichten eingesetzt wird muss vor dem Angriff die Stromversorgung der Lampe getrennt werden, damit der Angriff Erfolg hat. [33]

4.5.3 ZigBee Sniffing und Capturing

Im Paper von Olawumi et al. aus dem Jahr 2014 werden drei praktische Angriffe auf ZigBee-Netzwerke gezeigt. Beim ersten Angriff steht das Erkennen möglicher Angriffsziele im Vordergrund. Dabei werden verfügbare ZigBee-Geräte ermittelt und in weiterer Folge die Identität dieser im Netzwerk bestimmt. Als zweiter Angriff wird das Aufzeichnen von ZigBee-Traffic beschrieben. Abschließend wird für den dritten Angriff das Killerbee Framework verwendet, um die aufgezeichneten Pakete zu speichern.
Um das System vor dem ersten Angriff zu schützen, empfehlen sie die Verwendung eines Intrusion Detection Technik, mit dessen Hilfe die der Beacon Frame Prozess in ZigBee-Netzwerken überwacht werden sollte. Um sich vor dem zweiten Angriff zu schützen empfehlen Olawuma et al. die Verwendung des High-Security Levels in sicherheitskritischen ZigBee-Netzwerken. [2]

4.5.4 Angriffe auf die Batterie

Folgend werden zwei verschiedene Angriffe auf die Batterie von Geräten beschrieben, welche in weiterer Folge zur Ausnutzung durch einen Replay Angriff verwendet werden können.

ZigBee End Device (ZED) Sabotage

Bei dieser Art von Angriff wird versucht die Batterie eines Endgeräts möglichst schnell zu entleeren, indem der/die Angreifer*in vorgibt ein ZigBee-Router oder Koordinator zu sein und immer wieder Befehle an das Gerät sendet. So wird dieses davon abgehalten in den Sleep Modus zu wechseln. Im Normalbetrieb sendet ein ZigBee-Gerät einen Poll Request an seinen Parent, wenn es von seinem Sleep Zustand erwacht. Falls Daten für das Endgerät zur Verfügung stehen, antwortet der Parent mit einem Response sowie den Daten. Bei diesem Angriff antwortet der/die Angreifer*in via Broadcast und Multicast auf alle empfangenen Poll Requests. Dadurch wechseln die ZEDs nie mehr in den Sleep Modus, wodurch deren Batterie sehr schnell verbraucht wird. [34]

Energy Depletion Attack

Dieser Angriff zielt ebenso darauf ab, die Batterie eines ZigBee-Gerätes möglichst schnell zu entleeren damit sich dessen Frame Counter zurücksetzt. Da dieser unverschlüsselt übertragen wird ist es dem/der Angreifer*in möglich diesen auszulesen. So können Pakete mit einem höheren Frame Counter generiert werden, wodurch das Gerät das Paket nicht sofort beim Empfangen verwerfen wird. Das schadhafte Paket wird mit einem zufälligen Schlüssel verschlüsselt, wodurch die Entschlüsselung durch das Endgerät fehlschlägt. Auf Grund des fehlenden Schlüsselmaterials geht es hierbei auch nicht darum einen gültigen Befehl an das Gerät zu schicken, sondern lediglich dessen benötigte Rechenleistung in die Höhe zu treiben um so den Batterieverbrauch zu erhöhen. [35]

Um einen umfassenden Einblick in großteils theoretische Angriffe auf ZigBee-Netzwerke zu bekommen sind folgende Arbeiten empfehlenswert:

- IoT Device (ZigBee) Security Study [11]
- ZigBee Security Vulnerabilities: Exploration and Evaluating [15]
- New LDoS Attack in Zigbee Network and itsPossible Countermeasures [36]
- Detecting Spoofing Attacks in Zigbee using Device Fingerprinting [37]
- WazaBee: attacking Zigbee networks by diverting Bluetooth Low Energy chips [38]

5 ZigBee Network Intrusion Detection System

In Kapitel 4 wurde festgestellt, dass die untersuchten Angriffe bestimmte Muster aufweisen. Darauf aufbauend wird folgend das entwickelte System beschrieben, welches es ermöglicht diese zu erkennen. Dieses ermöglicht neben der automatisierten Erkennung auch die Benachrichtigung einer hinterlegten Mailadresse. Dabei wird auf den entwickelten Prozess und die Funktionsweise des Systems sowie das zugrunde liegende Datenbankdesign eingegangen. Ebenso werden die, in Kapitel 4 gelisteten, Funktionsabläufe der Angriffe um das Vorhandensein des Network Intrusion Detection Systems (NIDS) erweitert. Dadurch wird auch grafisch veranschaulicht, welche Vorteile dessen Verwendung mit sich bringt. Das System muss dazu nicht an das individuelle ZigBee Netzwerk angepasst werden. Lediglich die Mailadresse, welche im Falle eines Angriffs benachrichtigt werden soll, sollte eingetragen werden.

5.1 Motivation

Wie in den vorhergehenden Kapiteln gezeigt ist es relativ einfach ZigBee-Geräte anzugreifen oder sogar zu übernehmen. Während meiner Recherche bin ich auf keine Arbeit gestoßen, welche alle genannten Angriffe erkennt. Deshalb wurden Muster aus verschiedenen Angriffen abgeleitet und in weiterer Folge ein System zur Erkennung dieser entwickelt.
Das im Zuge dieser Diplomarbeit entwickelte NIDS ermöglicht es typische Angriffe auf ZigBee-Netzwerke zu erkennen und die Benutzer*innen zu benachrichtigen. Dieses System funktioniert herstellerunabhängig, wodurch der Einsatz für alle ZigBee-Benutzer*innen einen Mehrwert darstellt. Ein weiterer Vorteil ist die einfache Integration in eine bestehende Umgebung, wobei hierfür lediglich ein paar Abhängigkeiten installiert und das Projekt aus dem GIT Repository[1] heruntergeladen werden muss. Nach wenigen lokalen Einstellungen wie Mailadresse und Datenbankverbindung kann das System verwendet werden. Da zu Beginn des Prozesses der Input in ein eigenes Objekt konvertiert wird kann auch relativ einfach die, für das Aufzeichnen des Traffics verwendete, Quelle ausgetauscht werden. Des Weiteren funktioniert das entwickelte System autark und ist somit nicht aus dem Internet erreichbar und kann auch ohne Internetverbindung betrieben werden. In zweiterem Fall steht die Funktionalität der Benachrichtigung

[1] https://git.nwt.fhstp.ac.at/is201812/ZigBeeNIDS

per Mail nicht zur Verfügung. Dies kann aber gegebenenfalls durch Änderungen im Code auf eine lokale Benachrichtigungsmethode erweitert werden.
Der Angriff, in dem T. Zillner ein Türschloss öffnet, ohne dass der/die Besitzer*in dies bemerkt, kann zwar nicht verhindert werden. Der große Vorteil wäre aber, dass das System diesen Vorgang als Angriff erkennt und danach automatisiert den/die Besitzer*in per Mail benachrichtigt.

5.2 Komponenten eines ZigBee-Netzwerks

In Abbildung 5.1 werden die Komponenten, welche bei der Verwendung von ZigBee miteinander interagieren, dargestellt. Dabei wird prinzipiell zwischen IP Traffic und ZigBee-Traffic unterschieden. In der folgenden Abbildung bedient der/die Benutzer*in mittels APP die ZigBee-Komponenten. Dabei befindet sich das Smartphone im WLAN, wodurch der Router die Befehle an den ZigBee-Koordinator, welcher in diesem Fall das Trust Center darstellt, weiterleitet. Das entwickelte ZigBee NIDS ist ebenfalls mit dem Internet verbunden, um im Bedarfsfall die Besitzer*innen per E-Mail benachrichtigen zu können. Dabei kann in der Datenbank hinterlegt werden zu welchen Angriffen Benachrichtigungen erhalten werden wollen. Die Kommunikation zwischen den ZigBee-Komponenten sowie dem NIDS und dem/der Angreifer*in findet ausschließlich über das Protokoll ZigBee statt. Das NIDS hat die bereits öffentlich bekannten Master Keys hinterlegt. Der aktuelle Netzwerk-Key hingegen muss dort nicht hinterlegt werden. Somit bietet ein Angriff auf das NIDS für einen/eine Angreifer*in keinen Mehrwert, da dieses punkto ZigBee keine Informationen enthält, welche diesem/dieser nicht ohnehin bereits zur Verfügung stehen. In dem aufgebauten Testszenario wurde diverse Beleuchtungskomponenten miteinander verbunden. Die Hardware des entwickelten Systems bildet ein Raspberry PI 4 sowie ein Conbee II, um ZigBee-Traffic aufzeichnen zu können. Um die empfangenen Signale verarbeiten zu können werden diese mittels deConz an Wireshark weitergeleitet und in weiterer Folge mit dem python Framework pyshark interpretiert. Das System wurde zur Gänze selbst entwickelt und ist in meinem Git-Repository öffentlich zugänglich. Um Angriffe durchführen zu können wurde ein Attify ApiMote v4 Beta verwendet, welches als ZigBee Pentesting Toolkit fungiert. Zur Durchführung der Angriffe wird das weit verbreitete Framework namens Killerbee [39] verwendet.

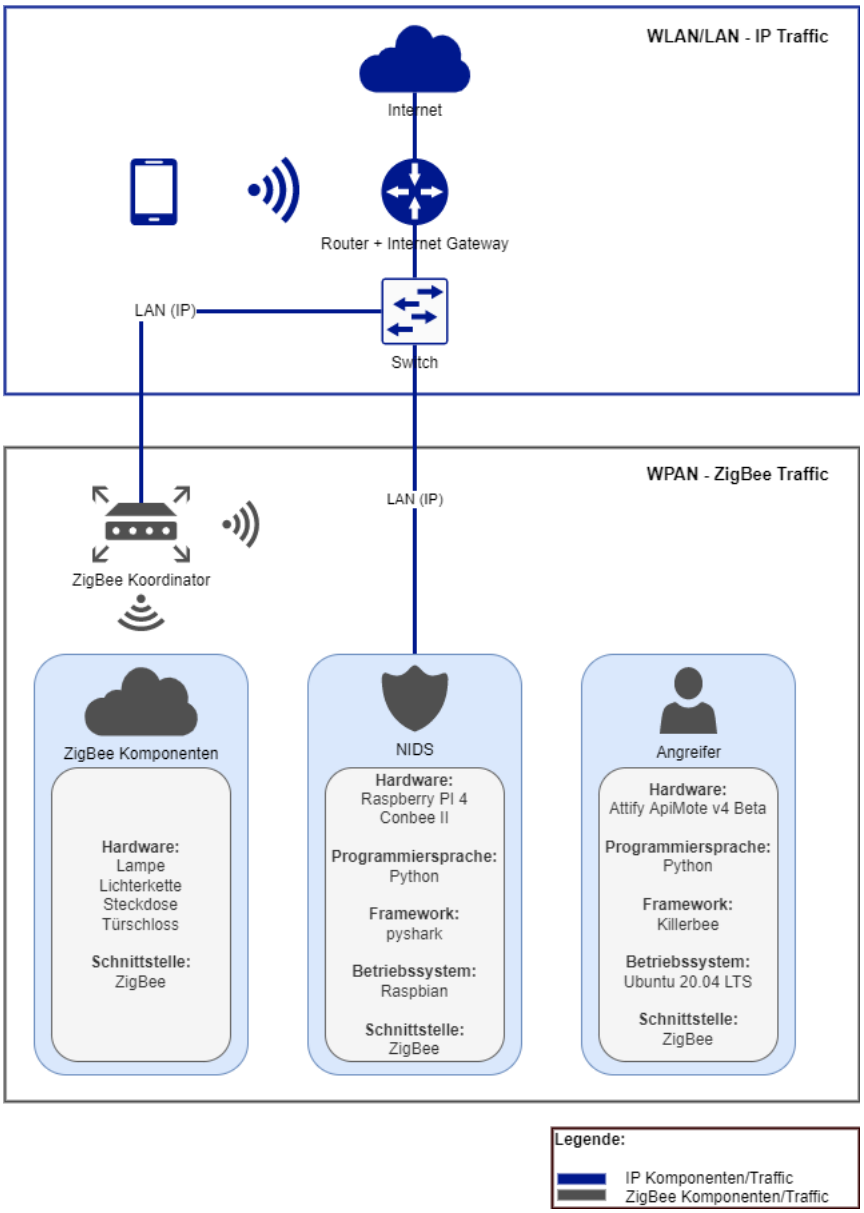

Abbildung 5.1: Komponenten des NIDS

5.3 Prozess des NIDS

Dieses Kapitel wird den Prozess des entwickelten Systems erklären. Dabei durchläuft jedes ZigBee-Paket diesen Prozess und wird anschließend in die Datenbank gespeichert. In Abbildung 5.2 werden die einzelnen Prozessschritte des NIDS detailliert erklärt. Wie bereits erwähnt werden als Hardware ein Raspberry PI 4 und ein ConBee II USB-Stick verwendet, um den ZigBee Traffic aufzeichnen zu können. Die Software zshark wird vom Hersteller der Sticks zur Verfügung gestellt und bildet die Verbindung zwischen ConBee II und Wireshark. Die aufgezeichneten Pakete können dabei entweder lokal mit Wi-

reshark weiterverarbeitet oder an einen entfernten Wireshark Server gesendet werden. Wireshark dient hierzu lediglich zur Visualisierung des aktuellen Traffics und muss für die Funktion des NIDS nicht zwingend gestartet sein. In Wireshark sind die öffentlich bekannten ZigBee Default Keys hinterlegt. Somit hat das NIDS die gleiche Sicht auf den ZigBee-Traffic wie mögliche Angreifer*innen, da alle Pakete, die diese entschlüsseln könnten mit Hilfe der Default Keys entschlüsselt werden. Auch wenn man die Angreifer*innen bereits innerhalb des Netzwerks vermutet, werden die implementierten Angriffstypen vom NIDS erkannt. Es wurde bei der Entwicklung darauf geachtet, dass diese unabhängig von der Phase des Angriffs erkannt werden. Des Weiteren wurde das NIDS so umgesetzt, dass der eigene Netzwerk-Key hierfür nicht hinterlegt werden muss. Dadurch wird sichergestellt, dass kein zusätzliches Angriffsziel für potenzielle Angreifer*innen geschaffen wird. Im nächsten Schritt werden die einzelnen Pakete mit der python-Library pyshark als Capture-Stream ausgelesen. Auf Grund der Eigenschaften von python kann nicht sichergestellt werden, dass jedes Capture Paket gleich aufgebaut ist. Aus diesem Grund ist der nächste Schritt die Normalisierung von diesem in eine eigens definierte Klassenstruktur. Dabei wird ein Objekt erstellt und mit den verfügbaren Werten des Capture Pakets befüllt. Ab diesem Zeitpunkt verwendet das NIDS nur noch das eigene Objekt. So kann sichergestellt werden, dass es zu keinem Laufzeitfehler auf Grund einer falschen Objektstruktur kommen kann. Ein weiterer Vorteil dieser Lösung ist, dass dadurch einfach die Input Quelle geändert werden kann, ohne das ganze Programm neu entwickeln zu müssen. Sollte also beispielsweise in einer späteren Arbeit der ConBee II durch andere Hardware ersetzt werden, so muss lediglich das Mapping des Captures in das definierte ZigBeePaket-Objekt angepasst werden. Im nächsten Schritt wird das ZigBeePaket-Objekt gegen die Muster in der Datenbank geprüft um festzustellen, ob es sich hierbei um einen Angriff handelt. Eine Kombination mehrerer Angriffe pro Paket ist nicht sinnvoll, wodurch aus Performancegründen schon beim Erkennen des Ersten eine Benachrichtigung versendet wird. Folgend werden die fünf implementierten Prüfungen erklärt, welche in weiterer Folge in Abschnitt 5.5 detailliert beschrieben werden:

- checkResetToFactory: Hierbei wird das ZigBee-Paket auf das Auftreten eines ResetToFactory Befehls überprüft.
- checkInsecureRejoin: Sollte es sich bei dem Paket um einen erfolgreichen Insecure Rejoin Response handeln wird in der Tabelle tbZigBeeData nach dem zugehörigen, zuvor aufgezeichneten, Request gesucht. Ist dieser vorhanden hat ein Insecure Rejoin stattgefunden.
- checkReplay: Hierbei wird die Datenbank dahingehend überprüft, ob dasselbe Paket zuvor schon einmal aufgezeichnet wurde. Dabei werden die Zieladresse (destaddress), der Message Integrity Code (zbee_sec_mic), der Frame Counter (zbee_sec_counter) sowie die Sequence Number

(nwk_seqno) auf Gleichheit überprüft. Da im Normalbetrieb dasselbe Paket nicht mehrmals übertragen wird kann man bei dessen Vorkommen davon ausgehen, dass es sich um einen Replay Angriff handelt. Die eigentlichen Nutzdaten beziehungsweise der übertragene Befehl werden nicht gespeichert, da diese für die Erkennung unerheblich sind. Da die Daten verschlüsselt beim NIDS ankommen und somit nicht interpretiert werden können würden diese nur unnötigen Speicherplatz in der Datenbank verbrauchen.

- checkTransportKey: Hierbei wird beim aktuellen Paket geprüft, ob es sich um ein Transport-Key Kommando handelt, welches den Netzwerk-Key übermittelt. Der Wireshark übernimmt, wie bereits erwähnt, zuvor die Entschlüsselung durch die Default-Keys. Wird die Übertragung des Netzwerk-Keys festgestellt kann davon ausgegangen werden, dass potenzielle Angreifer*innen diese ebenso aufzeichnen und entschlüsseln können.

- checkTouchlinkCommissioning: In diesem Fall wird geprüft, ob das Paket einen ”Network Join Request” enthält. Dies würde bedeuten, dass soeben versucht wird ein Gerät mittels Touchlink Commissioning in ein Netzwerk zu integrieren. Hierbei wird überprüft ob es sich bei dem beitretenden Gerät um einen Router oder ein Endgerät handelt.

Im nächsten Schritt wird das ZigBeePaket in die lokale Datenbank gespeichert. Ist ein Angriff aufgetreten oder handelt es sich um einen Insecure Rejoin Request so wird dieser Umstand bei dem Datenbankeintrag vermerkt. Des Weiteren wird dann geprüft ob für diesen Angriffstyp die Benachrichtigungen in der Datenbank aktiviert sind. Somit können die Benutzer*innen zur Laufzeit des Programms festlegen, über welche Angriffe sie informiert werden möchten. So kann beispielsweise die Benachrichtigung für die Übertragung des Netzwerk-Keys (checkTransportKey) temporär deaktiviert werden, falls eine Vielzahl an Geräten zum Netzwerk hinzugefügt wird. Sind Benachrichtigungen für diese Art von Angriff aktiv wird die hinterlegte Mailadresse über das Auftreten von diesem informiert. Nach dem Versenden der Mail wird das nächste Paket aus dem Stream gelesen und in weiterer Folge zu einem ZigBeePaket-Objekt normalisiert. Falls es sich um keinen Angriff handelt, wird wie in Abbildung 5.2 ersichtlich das nächste Paket normalisiert und geprüft.

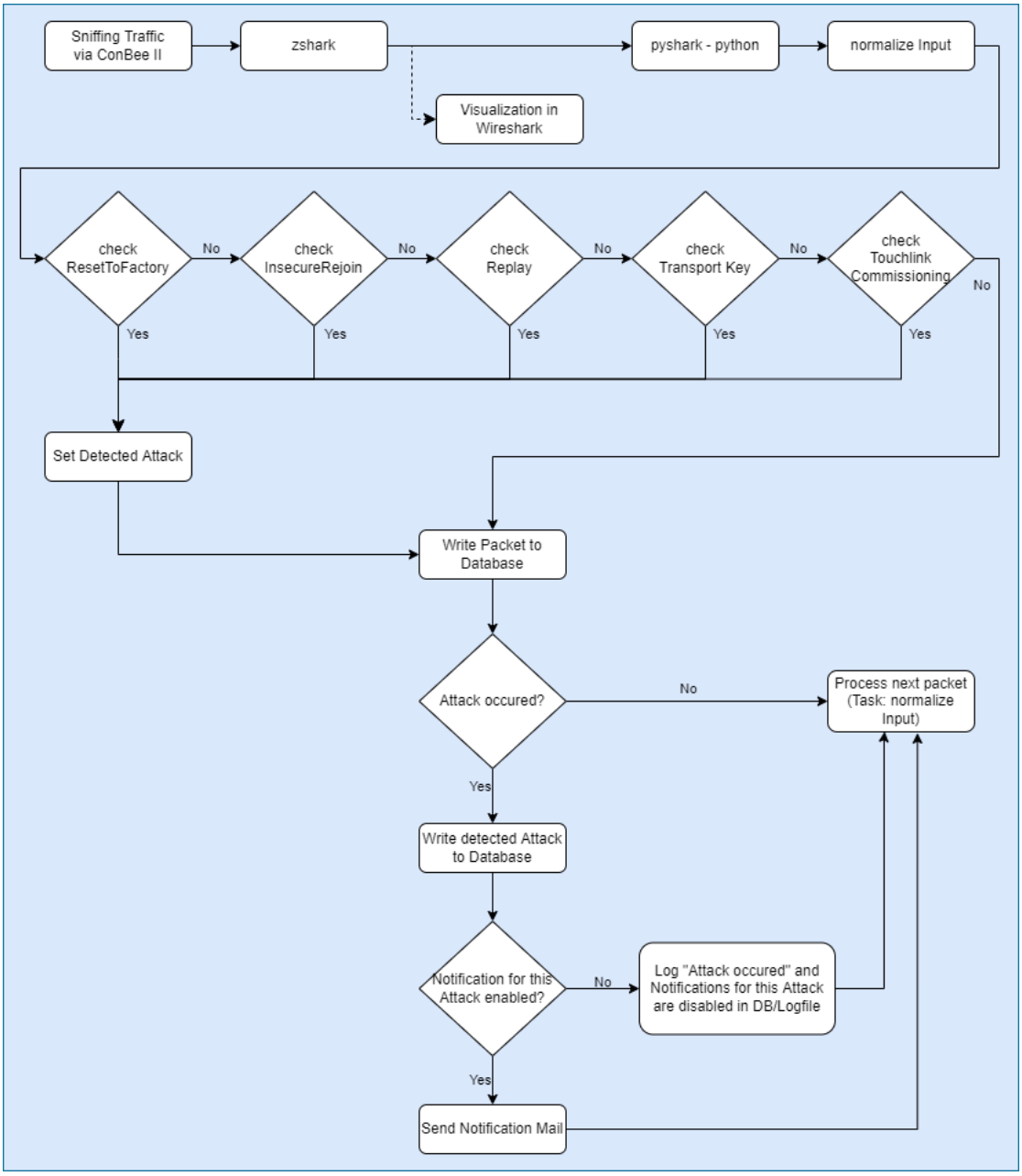

Abbildung 5.2: Prozess des NIDS

5.4 Integration in ein bestehendes ZigBee Netzwerk

Die Integration des Systems in ein bereits bestehendes ZigBee-Netzwerk ist relativ einfach. Das NIDS kann dazu sowohl auf einem bereits bestehenden als auch auf einem dafür angeschafften System installiert werden. Neben python3 und mariadb gilt es nur wenige Abhängigkeiten zu installieren. Nachdem der python Code aus dem GIT Repository geladen wurde, kann auch schon mit der Konfiguration der Datenbankverbindung begonnen werden. Dazu müssen in das lokale Konfigurationsfile die Zugangsdaten

zur Datenbank eingetragen werden. Danach kann die gesamte Datenbankstruktur mit einem vorgefertigten Script automatisiert erstellt werden woraufhin das NIDS sofort einsatzbereit ist. Eine detaillierte Anleitung zur Installation der Abhängigkeiten und dem Einrichten der Datenbank kann Abschnitt 5.9 entnommen werden.

5.4.1 Parametrisierung

Um das System leicht in bestehende Umgebungen integrieren zu können wurden alle notwendigen Einstellungen in einem zentralen Konfigurationsfile beziehungsweise der Datenbank selbst gespeichert. Somit sind bei der Installation und Inbetriebnahme keine Änderungen am Sourcecode notwendig. Die Einstellungen, welche sich in der Datenbank befinden, werden initial mit Default Werten erstellt und können entsprechend der eigenen Umgebung angepasst werden.
Um eine bessere Übersicht zu gewährleisten, finden sich sämtliche Konstanten in der Klasse Constants im gleichnamigen python File. Dort sind auch die Pfade zu Konfigurations- und Logfile hinterlegt. Wird ein Angriff erkannt wird zusätzlich zum Eintrag in der Datenbank ein Eintrag in das Logfile getätigt.

5.4.2 Logging

In erster Linie wurde das Logging im Zuge vom Errorhandling in die Applikation integriert. Diese Funktionalität wurde dahingehend erweitert, dass auch die erkannten Angriffe zusätzlich im Logfile festgehalten werden. Ebenso kann diesem im Nachhinein entnommen werden, ob zum Zeitpunkt des Angriffs die Benachrichtigung für diesen Typ aktiv war. Im folgenden Snippet wird dieser Unterschied anhand von Beispielen gezeigt.

```
2021-07-27 07:30:50 AM;INFO;Replay detectet
2021-07-27 07:30:51 AM;INFO;Notification Email sent successfully!

2021-09-30 01:13:28 PM;INFO;ResetToFactory detectet
2021-09-30 01:13:28 PM;INFO;Notification for this Thread is disabled
                           in Database!
```

5.5 Implementierte Angriffserkennung

Die vier bereits definierten Angriffstypen werden vom NIDS erkannt und lauten wie folgt:

- Insecure Rejoin

- Replay
- Übernahme von Geräten
- Netzwerk-Key Sniffing

5.5.1 Insecure Rejoin

Wie in Abschnitt 4.1 gezeigt werden bei einem Insecure Rejoin der NWK Rejoin Request sowie der NWK Rejoin Response unverschlüsselt übertragen. Dieser Vorgang wird vom System erkannt und je nach Konfiguration erfolgt automatisiert eine Benachrichtigung.
Das System erkennt dabei einen unverschlüsselten NWK Rejoin Request und speichert diese Information in der Spalte ”isInsecureRejoinRequest” der Datenbanktabelle ”tbZigBeeData”. Ebenso wird bei allen ZigBee-Paketen geprüft, ob es sich um einen unverschlüsselten NWK Rejoin Response handelt. Beim Erkennen eines solchen Pakets wird im nächsten Schritt die Gültigkeit dieser Antwort überprüft. Dazu wird in der Datenbank kontrolliert, ob zuvor ein NWK Rejoin Request stattgefunden hat. Stimmt die Zieladresse des aufgezeichneten Rejoin Response mit der Quelladresse des Rejoin Requests überein so wird dies als Insecure Rejoin erkannt. Abschließend wird das Paket in die Tabelle ”tbZigBeeData” gespeichert. Hierbei wird der Datensatz um die Information ergänzt, dass ein Angriff erkannt wurde. Je nach Konfiguration erfolgt abschließend die Alarmierung per E-Mail.

Ablauf des Angriffs bei Verwendung des NIDS

In Abbildung 5.3 wird der Ablauf des Angriffs inklusive der zugehörigen Funktionsweise des NIDS beschrieben.

1. Der/Die Angreifer*in stört die Kommunikation zwischen dem ZigBee-Gerät und dessen Parent.
2. Da das ZigBee-Gerät nicht mehr antwortet entfernt der Parent dieses aus dem Netzwerk.
3. Das ZigBee-Gerät sendet einen unverschlüsselten NWK Rejoin Request, um dem Netzwerk erneut beizutreten.
4. Das Network Intrusion Detection System (NIDS) erkennt diesen Request und speichert diese Information zu dem Paket in die Tabelle ”tbZigBeeData”.
5. Der Parent sendet einen NWK Rejoin Response an das ZigBee-Gerät.
6. Das NIDS erkennt den Response und prüft ob es einen zugehörigen Request in der Datenbank gibt.

7. Falls es diesen gibt, erfolgt je nach Konfiguration die Benachrichtigung der hinterlegten Mailadresse.

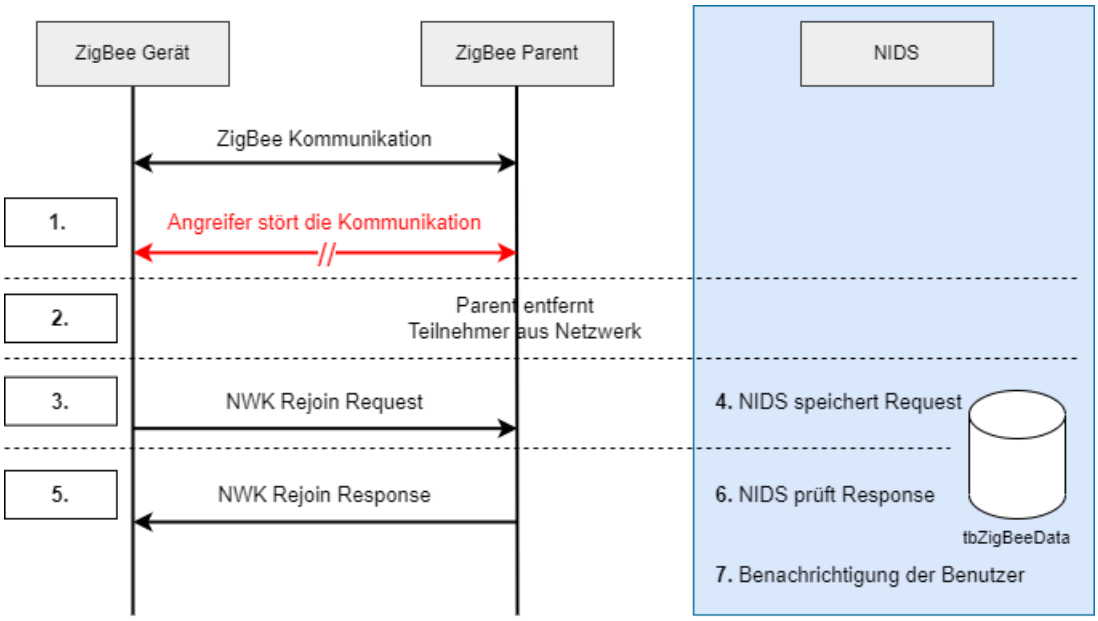

Abbildung 5.3: Ablauf Insecure Rejoin bei Verwendung vom NIDS

5.5.2 Replay

Wie bereits in Abschnitt 4.3 beschrieben handelt es sich bei einem Replay Angriff um das erneute Übermitteln von zuvor aufgezeichneten Nachrichten. Um unnötigen Overhead zu vermeiden, wird nicht das gesamte Paket in die Datenbank gespeichert, sondern lediglich dafür relevante Felder. Für Replay Angriffe sind vor allem diese Felder eines ZigBee-Pakets interessant:

- nwk.dst oder wpan.dst64: Empfänger Adresse des ZigBee-Pakets
- zbee_sec_mic: Message Integrity Code
- zbee_sec_counter: Frame Counter
- nwk_seqno: Sequence Number

Durch die Kombination aus MIC, Frame Counter und Sequence Number kann ein Paket eindeutig identifiziert werden.

Ablauf des Angriffs bei Verwendung des NIDS

In Abbildung 5.4 wird der Ablauf eines Replay Angriffs, welcher in Abschnitt 4.3 detailliert beschrieben wurde, um die Funktionalität des entwickelten NIDS erweitert.

1. Initial wird die ZigBee-Kommunikation von dem/der Angreifer*in aufgezeichnet. Hierbei handelt es sich beispielsweise um den ”Tür öffnen” Befehl in verschlüsselter Form. Ebenso wird dieses Paket auch vom NIDS aufgezeichnet und in die Datenbank gespeichert.

2. Im nächsten Schritt bringt der/die Angreifer*in das Gerät dazu sich neu zu starten oder auf Werkseinstellungen zurückzusetzen, sodass der Frame Counter neu initialisiert wird.

3. Der/Die Angreifer*in übermittelt nun die zuvor aufgezeichneten, immer noch verschlüsselten, ZigBee-Pakete erneut an das Zielgerät. Dieses wird den Befehl akzeptieren und ausführen.

4. Das NIDS verarbeitet die von dem/der Angreifer*in übermittelten Pakete und prüft in der Datenbank ob in der Vergangenheit schon ein Paket mit identischer Empfänger Adresse, Message Integrity Code, Frame Counter und Sequence Number übertragen wurde.

5. Je nach Konfiguration erfolgt die Benachrichtigung an die Mailadresse über das Auftreten eines Replay Angriffs.

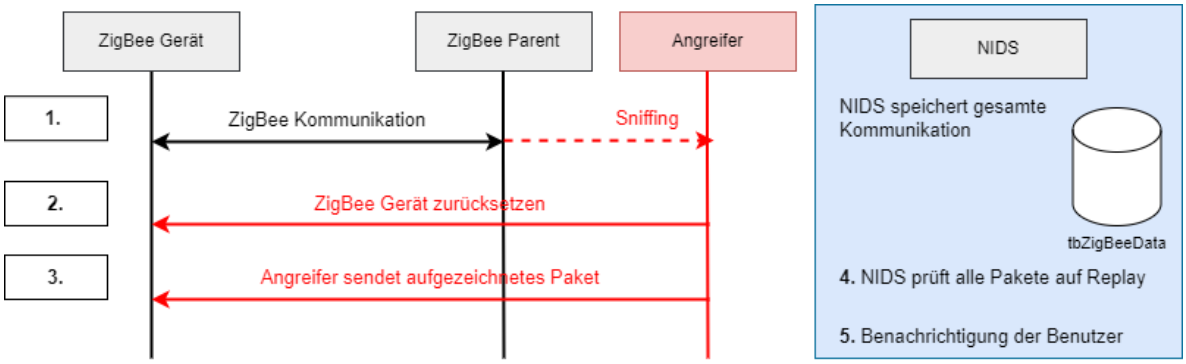

Abbildung 5.4: Ablauf Replay bei Verwendung vom NIDS

5.5.3 Übernahme von Geräten

Wie bereits in Abschnitt 4.2 erklärt muss ein ZigBee-Gerät dazu gebracht werden nach verfügbaren Netzwerken zu suchen, um dieses übernehmen zu können. Eine sehr einfache Möglichkeit um dies zu erreichen ist das Senden des ”Reset to Factory” Befehls, weswegen das System auch Pakete dahingehend überprüft. Beinhaltet ein ZLL Paket diesen Befehl, werden je nach Konfiguration die Benutzer*innen benachrichtigt. Unabhängig davon wird überprüft, ob es sich bei dem Paket um einen ”Network Join Request” handelt. Um unterscheiden zu können ob es sich um einen Beitritt zum Netzwerk der Benutzer*innen oder jenes der Angreifer*innen handelt ist es möglich die eigene PAN-ID in der ”tbGeneralSettings” unter dem Key ”PANID” zu hinterlegen. Unterscheidet sich die eingetragene PAN-ID von

jener des aktuellen Pakets deutet dies darauf hin, dass Angreifer*innen versuchen, ein oder mehrere ZigBee-Geräte zu übernehmen. Wird keine PAN-ID hinterlegt, so wird diese auf Grund der fehlenden Unterscheidungsmöglichkeit nicht auf Gleichheit überprüft. In diesem Fall wird bei jedem ”Network Join Request” ein potenzieller Angriff erkannt.

Ablauf des Angriffs bei Verwendung des NIDS

In Abbildung 5.5 ist der Ablauf eines solchen Angriffs inklusive Erkennung durch das NIDS grafisch aufbereitet.

1. Der/Die Angreifer*in sendet einen ”Reset To Factory” Befehl an ein ZigBee-Gerät.
2. Das System erkennt diesen und benachrichtigt die Benutzer*innen darüber, dass versucht wird ein Gerät auf Werkseinstellungen zurückzusetzen.
3. Das Gerät setzt sich auf Werkseinstellungen zurück und ist nun bereit einem neuen Netzwerk beizutreten.
4. Der/Die Angreifer*in sendet einen Network Join Request an das Gerät.
5. Das NIDS erkennt diesen Request und prüft zusätzlich dessen PAN-ID gegen jene, die in der Datenbank hinterlegt wurde. Weichen diese voneinander ab handelt es sich bei dem Beitritt um ein anderes Netzwerk. Falls die PAN-ID identisch ist, handelt es sich um einen Beitritt zum Netzwerk der Benutzer*innen. Wurde diese nicht in der Datenbank hinterlegt, wird diese Prüfung übersprungen und ein potenzieller Angriff erkannt.
6. Der Client tritt dem Netzwerk von dem/der Angreifer*in bei.
7. Je nach Konfiguration erfolgt die Benachrichtigung der Benutzer*innen.

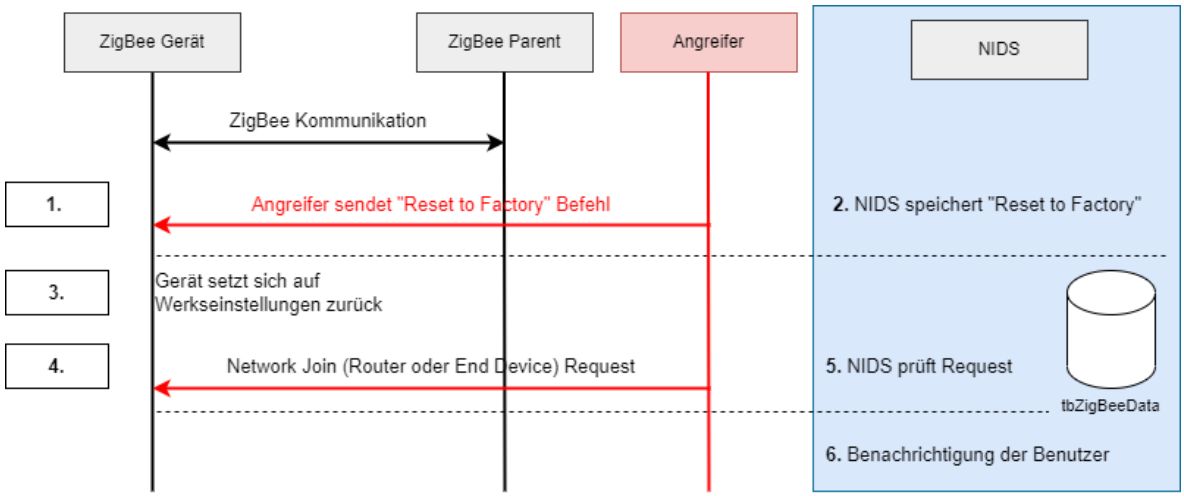

Abbildung 5.5: Ablauf Übernahme von Geräten bei Verwendung vom NIDS

5.5.4 Netzwerk-Key Sniffing

In Abschnitt 4.4 wird erklärt wie der Angriff bei der Erstübertragung des Netzwerk-Keys funktioniert. Daraus resultierend ergibt sich für das NIDS die Aufgabe einen, mit öffentlich bekannten Default Key verschlüsselten, Schlüsseltausch zu erkennen. Bei der Inbetriebnahme einer Vielzahl an neuer Geräte könnte man deswegen die Benachrichtigungen für diesen Angriff in der Datenbank temporär deaktivieren um eine Flut an Nachrichten zu vermeiden. Auf der anderen Seite ist dies ein gute Möglichkeit um die Funktionalität des NIDS zu testen.

1. Network Layer Management Entity (NLME) Services werden verwendet um einen Request für den Beitritt zum Netzwerk zu stellen.
2. Der Router leitet die Anfrage an das Trust Center weiter, wobei dieses prüft ob das Gerät dem Netzwerk beitreten darf.
3. Die Antwort vom Trust Center wird verschlüsselt an den Router gesendet.
4. Der Router übermittelt den Netzwerk-Key (mit DTCLK verschlüsselt) an das ZigBee-Gerät. Der/-Die Angreifer*in kann diese Übertragung aufzeichnen und mit dem Default Trust Center Link Key entschlüsseln.
5. Das NIDS bemerkt die Übertragung vom Netzwerk-Key falls diese mittels Default Trust Center Link Key, oder einem der anderen Default Keys, verschlüsselt wird.
6. Je nach Konfiguration erfolgt die Benachrichtigung der Benutzer*innen.

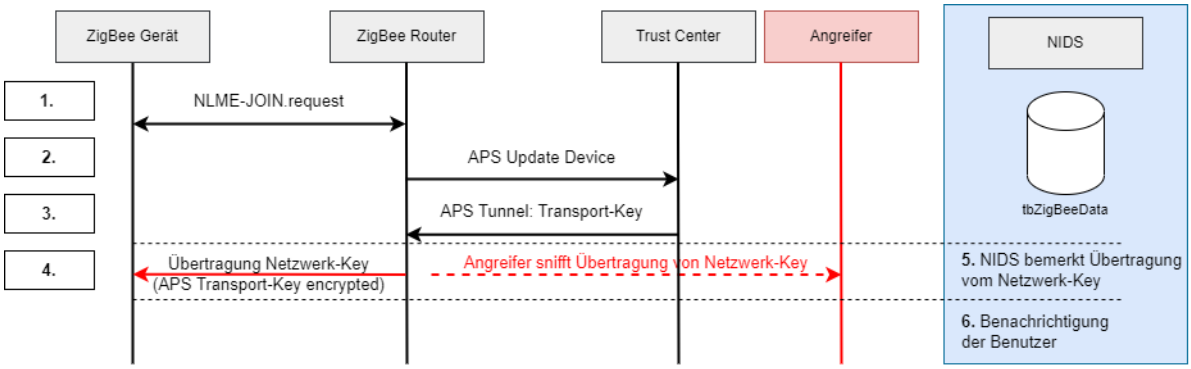

Abbildung 5.6: Ablauf Erstübertragung des Netzwerk-Keys bei Verwendung vom NIDS

5.6 Datenbankdesign

Um die Pakete persistent speichern zu können wird ein MariaDB Server verwendet. Hierbei handelt es sich um eine weit verbreitet Relationale Datenbank, welche Open Source verfügbar ist. [40] Das System wurde modular aufgebaut und es wurde vor allem auf eine leichte Erweiterbarkeit geachtet. Aus diesem Grund wurden die zu speichernden Informationen in fünf Tabellen unterteilt:

- tbZigBeeData

  Diese Tabelle beinhaltet alle für den Betrieb des NIDS notwendigen Informationen zu jedem aufgezeichneten Paket inklusive einer eigens generierten UUID, welche tabellenübergreifend als Identifier dient. Des Weiteren wird in dieser Tabelle gespeichert, ob es sich bei dem jeweiligen Paket um einen möglichen Angriff handelt.

- tbDataThreadMapping

  Alle erkannten Angriffe werden in dieser Tabelle gespeichert. Dies beinhaltet zum einen die UUID, um in weiterer Folge den Angriff einem Paket zuordnen zu können. Zum anderen wird die hinterlegte ID des Threads aus der Tabelle tbThreadSettings gespeichert. So ist es möglich im Nachhinein festzustellen welcher Angriffstyp durch welches Paket erkannt wurde.

- tbThreadSettings

  Hier sind alle implementierten Angriffe beziehungsweise Gefahren hinterlegt. Hierbei kann pro Angriffstyp festgelegt werden, ob im Falle des Auftretens eine Nachricht versendet werden soll. Diese Tabelle wird beim Erstellen der Datenbankstruktur automatisiert mit Default Werten befüllt. Hierbei werden standardmäßig für alle Angriffe Benachrichtigungen ausgelöst.

- tbGeneralSettings

In dieser Tabelle können generelle Informationen in Form von Key-Value Pairs gespeichert werden. Die Empfänger-Mailadresse für die Benachrichtigungen wird hier hinterlegt. Beispielsweise können die bekannten Default Keys ebenso in dieser Tabelle hinterlegt werden. Diese sind zwar für den Betrieb derzeit nicht notwendig, da die Pakete bereits entschlüsselt vom Wireshark abgeholt werden. Sollte man jedoch den Input so verändern, dass die Entschlüsselung in python mittels pyshark stattfinden soll, könnten diese Keys hier hinterlegt werden.

Die gesamte Datenbankstruktur kann automatisiert erstellt werden. Dazu gibt es innerhalb der Klasse DatabaseHandling.py die Funktion recreateDatabaseStructure(), welche etwaige bestehende Datenbankeinträge löscht und wieder mit Default Werten überschreibt.

5.6.1 Entity Relationship Design

In Abbildung 5.7 ist das Entity Relationship Modell des entwickelten NIDS abgebildet. Dieser können die Beziehungen zwischen den einzelnen Datenbanktabellen entnommen werden. Grundlegend verfügt jede Tabelle über eine eigene ID. Die generierte UUID der Tabelle tbZigBeeData wird in der Tabelle tbDataThreadMapping unter dem Spaltennamen tbZigBeeDataUUID als Foreign Key verwendet. Die Tabelle tbDataThreadMapping verfügt des Weiteren mit der Spalte tbThreadSettingsID über eine Referenz auf die gleichnamige Tabelle tbThreadSettings.

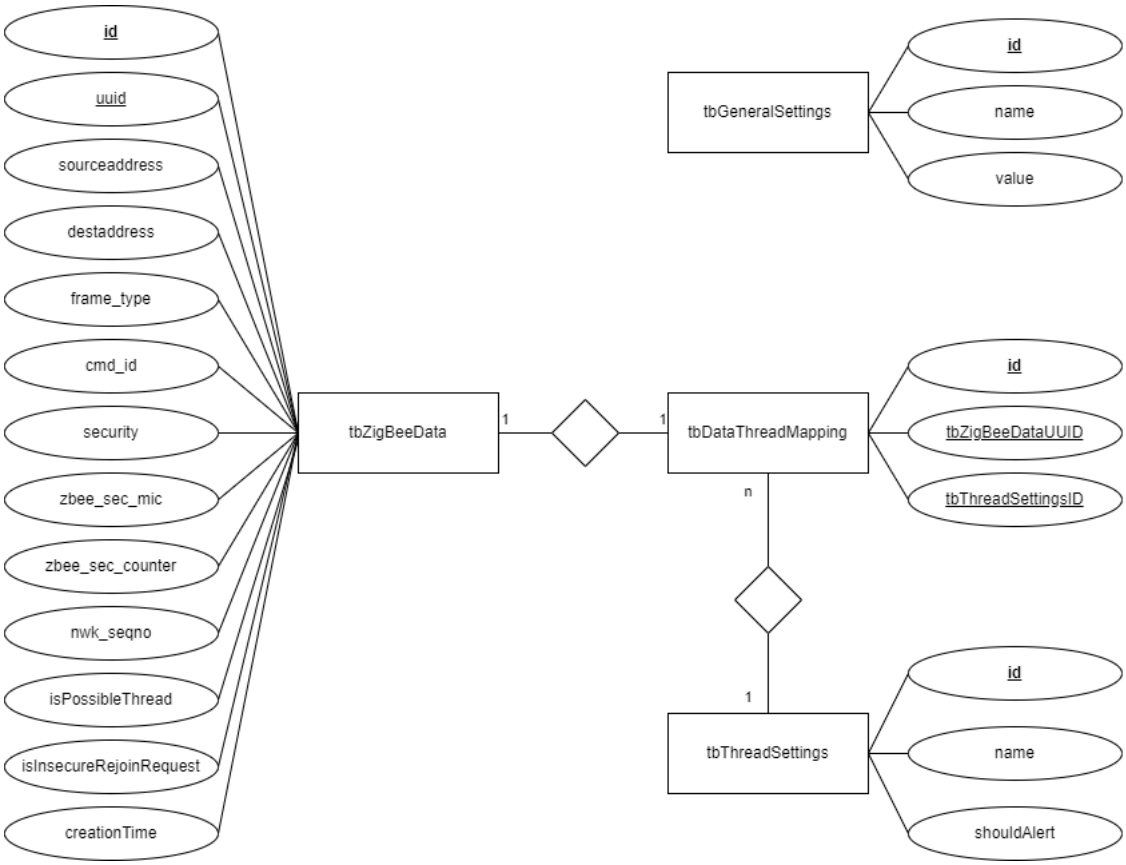

Abbildung 5.7: Entity Relationship Design

5.6.2 Relationships

tbZigBeeData - tbDataThreadMapping

1:1 Beziehung

Ein ZigBee-Paket kann nur maximal einem Angriff zugeordnet werden und ein aufgezeichneter Angriff kann nur von einem ZigBee-Paket stammen.

tbThreadSettings - tbDataThreadMapping

1:n Beziehung

Ein definierter Angriffstyp kann mehrmals erkannt werden, wohingegen ein aufgezeichneter Angriff nur einem bestimmten Angriffstyp zugeordnet werden kann.

5.6.3 Relationales Schema

In Abbildung 5.8 wird das Relationale Modell der entwickelten Datenbank dargestellt. Jedes neue Paket wird in die Tabelle tbZigBeeData gespeichert, wobei wie bereits erwähnt mittels uuid in der Tabel-

le tbDataThreadMapping darauf referenziert wird. In der Tabelle tbThreadSettings kann für sämtliche Angriffe hinterlegt werden ob dafür eine Benachrichtigung versendet werden soll. Die Tabelle tbDataThreadMapping beinhaltet alle erkannten Angriffe, wobei durch die beiden Foreign Keys Zugriff auf Details ermöglicht werden.

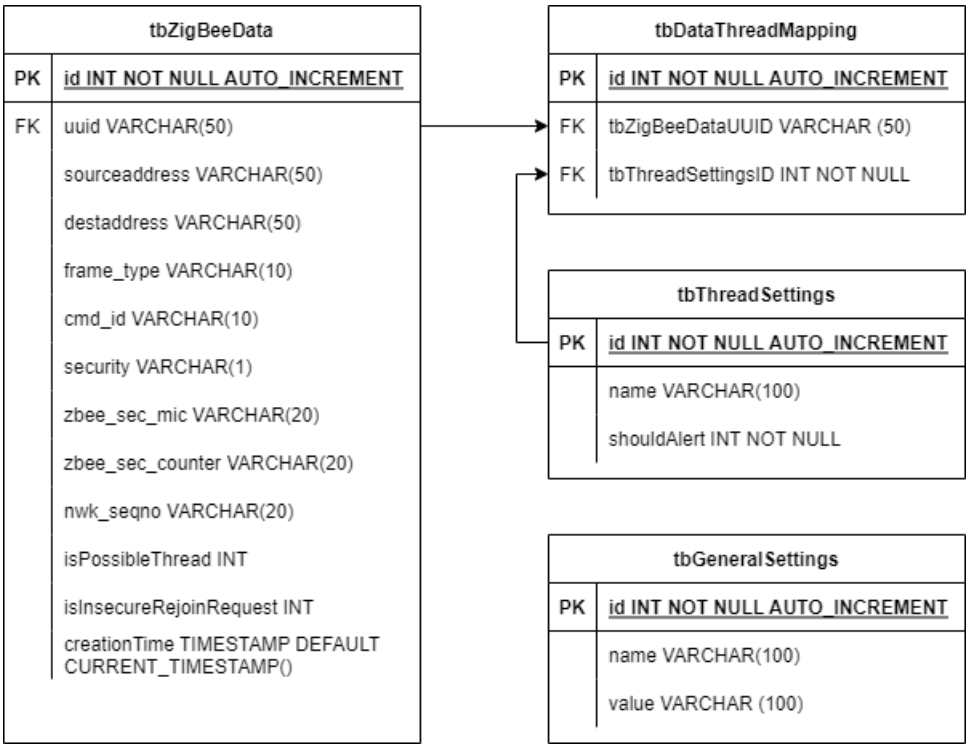

Abbildung 5.8: Relationales Modell

5.7 Felder des ZigBee-Pakets

Um Overhead zu vermeiden und python-Laufzeitfehlern vorzubeugen werden wie bereits erwähnt die aufgezeichneten ZigBee-Pakete zu Beginn des Prozesses in ein eigens erstelltes python-Objekt normalisiert. In Tabelle 5.1 sind die relevanten Felder inklusive deren Verwendungszweck aufgelistet. Dabei ist die erste Spalte der Spaltenname des Feldes in der Datenbank des NIDS. Einige Felder werden nur zur Laufzeit benötigt und deshalb nicht in der Datenbank gespeichert. ”Feld in ZigBee Header” beschreibt den originalen Namen des Feldes inklusive dessen Header aus dem originalen ZigBee-Paket. Bei Einträgen die mit ”||” getrennt angegeben sind wird je nach Anwendungsfall einer der beiden Werte verwendet. In der letzten Spalte der Tabelle ist die Funktion des Feldes angegeben.

Spalte in Datenbank	Feld in ZigBee Header	Funktion des Feldes im NIDS
uuid	-	Eindeutige ID des ZigBee Pakets innerhalb des NIDS
sourceaddress	zbee_nwk.src || wpan.src64	Quelladresse des ZigBee Pakets
destaddress	zbee_nwk.dst || wpan.dst64	Zieladresse des ZigBee Pakets
frame_type	zbee_nwk.frame_type	Typ des Frames wird bei der Erkennung von InsecureRejoin verwendet
cmd_id	zbee_nwk.cmd_id || zbee_zcl_general_touchlink_rx_cmd_id	ID des ZigBee Kommandos wird bei der Erkennung von InsecureRejoin, Netzwerk Key Sniffing, ResetToFactory, TouchlinkCommissioning verwendet
security	zbee_nwk.security	Security Flag wird bei der Erkennung von InsecureRejoin verwendet
zbee_sec_mic	zbee_nwk.zbee_sec_mic	Message Integrity Code wird bei der Erkennung von Replay verwendet
zbee_sec_counter	zbee_nwk.zbee_sec_counter	eindeutiger Frame Counter wird bei der Erkennung von Replay verwendet
nwk_seqno	zbee_nwk.seqno	Sequence Number wird bei der Erkennung von Replay verwendet
isPossibleThread	-	beschreibt ob es sich bei dem Paket um einen potentiellen Angriff handelt
isInsecureRejoinRequest	-	beschreibt ob es sich bei dem Paket um einen Insecure Rejoin Request handelt
creationTime	-	aktuelle Uhrzeit beim Aufzeichnen des Pakets
-	zbee_nwk.cmd_addr	Adresse innerhalb des Rejoin Response wird bei der Erkennung von InsecureRejoin verwendet
-	wpan.src_pan || wpan.dst_pan	Quell-PAN Adresse des ZigBee Pakets wird bei der Benachrichtigung verwendet
-	wpan.dst_pan	Ziel-PAN Adresse des ZigBee Pakets wird bei der Benachrichtigung verwendet
-	zbee_aps.cmd_id	ID des ZigBee Kommandos wird bei der Erkennung von Netzwerk Key Sniffing verwendet
-	zbee_nwk.cmd_rejoin_status	Rejoin Status des ZigBee Pakets wird bei der Erkennung von InsecureRejoin verwendet
-	zbee_nwk.src64	64-Bit Quelladresse des ZigBee Pakets wird als Quelladresse verwendet falls es keine NWK Quelladresse gibt
-	wpan.dst64	64-Bit Zieladresse des ZigBee Pakets wird als Zieladresse verwendet falls es keine NWK Zieladresse gibt
-	zbee_aps.cmd_key_type	Typ des Keys, der übertragen wird wird bei der Erkennung von Netzwerk Key Sniffing verwendet
-	wpan.pan_id_compression	ID die angibt, dass Source PAN und Destination PAN identisch sind wird bei der Normalisierung zum bestimmten der Source PAN ID verwendet

Tabelle 5.1: Normalisierte ZigBee Felder

5.8 Hardware

Als Hardware wird ein Raspberry PI 4 verwendet, welcher mittels ConBee II beziehungsweise RaspBee II um die Funktionalität des ZigBee-Traffic Aufzeichnens erweitert wird. Beim ConBee II handelt es sich um ein universelles ZigBee USB-Gateway der Firma dresden elektronik ingenieurtechnik gmbh, welches ohne Cloud arbeitet. Dieses funktioniert mit allen bekannten Home Automation Systemen und verfügt über eine hohe Reichweite durch Signalverstärker. In deren Produktpalette findet man ebenso den sogenannten RaspBee II, welcher direkt auf den Raspberry PI aufgesteckt werden kann. In diesem ist zusätzlich eine batteriebetriebene Echtzeituhr (RTC) integriert. Da in einem Raspberry PI eine solche per Default nicht verbaut ist kann somit sichergestellt werden, dass nach einem Stromausfall die gewünschten Tasks zur richtigen Zeit ausgeführt werden. Das NIDS kann mit beiden Produkten verwendet werden. [41]

Als Testumgebung werden Philips Hue Lichtsteuerungselemente verwendet. Diese bestehen aus verschiedenen Arten von Lampen, wobei zur Bedienung ein Dimmer Switch sowie ein Smartphone verwendet werden. Des Weiteren fungiert eine Philips Hue Bridge als zentrale Steuereinheit des Systems.

Um in weiterer Folge Angriffe auf das System durchführen zu können wird ein Attify ApiMote v4 Beta verwendet, wobei dies in Kapitel 6 detailliert beschrieben wird.

5.9 Entwicklung

Bevor mit der eigentlichen Entwicklung des Systems begonnen wurde startete die Evaluierung der Sniffing Hard- und Software, um die gewünschten Ergebnisse erzielen zu können. Dafür wurde im ersten Schritt mittels ConBee II die ZigBee-Channel gescannt, um erstmals Traffic visualisieren zu können. Als nächstes wurden der Default Trust Center Link Key, der Light Link Master Key sowie der Light Link Commissioning Key im Wireshark hinterlegt. So kann der Netzwerk-Key bei der initialen Übertragung mitgelesen werden, falls für diese einer der Keys zur Verschlüsselung verwendet wird. Wenn der eigene Netzwerk-Key in Wireshark eingetragen wird, erhält man einen detaillierteren Überblick über den Netzwerk-Traffic. Dies ist für die Funktionalität des NIDS jedoch nicht erforderlich.

Zur Verwaltung und Sicherung des Sourcecodes wurde das Versionierungssystem Git verwendet. Hierbei handelt es sich um eine weit verbreitete Open-Source Lösung, welche auch von der FH betrieben wird. Dazu wurde im ersten Schritt ein neues Projekt angelegt und auf die Entwicklungsumgebung geklont.[2]

Die relevanten Source Files wurden danach regelmäßig in das Git Repository gepushed. Ebenso findet sich im Repository eine Anleitung, welche die Verwendung des erstellten NIDS beschreibt. Als Pro-

[2] git clone https://git.nwt.fhstp.ac.at/is201812/ZigBeeNIDS.git

grammiersprache für das Network Intrusion Detection System wird python3 verwendet.

zshark stellt die Schnittstelle zwischen dem verwendeten ConBee II und Wireshark dar. Hierbei muss beim erstmaligen Verwenden des ConBee II die Firmware auf den Stick geflashed werden. Dies kann mittels zshark-GUI erledigt werden. Danach kann der gewünschte ZigBee-Kanal und die IP des Zielrechners, an den die aufgezeichneten Daten gesendet werden sollen, konfiguriert werden. Zshark kann auf den Plattformen Linux, Windows und Raspbian verwendet werden. Die aktuelle Version kann von der Website des Herstellers heruntergeladen werden.[3] Um zshark verwenden zu können ist noch das Paket *libqt5serialport5* erforderlich. Details zur Konfiguration finden sich in Abschnitt 9.2.
Die Verwendung von zshark ist simpel, da lediglich der ZigBee-Channel auf dem gelauscht werden soll sowie die IPv4 Adresse des Wireshark-Servers eingetragen werden muss. Dies ist die IPv4 Adresse vom lokalen eth Interface des Raspberry PI.

Im nächsten Schritt muss Wireshark installiert und entsprechend konfiguriert werden. Damit Wireshark in weiterer Folge Pakete auf dem Loopback Interface aufzeichnen kann muss es mit erhöhten Rechten gestartet werden. Damit nun auch ein non-root User Zugriff auf die aufgezeichneten Pakete im File */usr/bin/dumpcap* hat muss dieser entsprechend berechtigt werden. Um eine reibungslose Verwendung von zshark gewährleisten zu können muss in Wireshark noch ”Assume packets have FCS” und ”Validate the ethernet checksum if possible” deaktiviert werden. Die für die Konfiguration notwendigen Befehle finden sich in Abschnitt 9.3. Nachdem die grundlegenden Einstellungen getroffen wurden kann Wireshark gestartet werden. Dabei kann noch ein entsprechender Filter eingetragen werden um etwaigen Overhead rauszufiltern. Dieser lautet wie folgt:

```
udp.port==17754 && !icmp
```

Ein weiterer wichtiger Punkt ist das Eintragen, der bekannten Default Link-Keys in Wireshark. Somit wird der Traffic automatisch entschlüsselt, falls diese verwendet werden.

1. Default Global Trust Center Link Key
2. Light Link Master Key
3. Light Link Commissioning Key

pyshark ist ein python-wrapper für tshark, wodurch es möglich ist Netzwerktraffic in python zu verarbeiten. Dieses Paket kann sowohl unter Linux als auch unter Windows verwendet werden.[4]

[3] https://phoscon.de/downloads/zshark/
[4] https://github.com/KimiNewt/pyshark

tshark ist das Kommandozeilenprogramm von Wireshark, welches es ermöglicht Netzwerktraffic aufzuzeichnen und zu analysieren. Hierbei können die aufgezeichneten Pakete einerseits in das Wireshark-Standardformat pcap gespeichert werden. Andererseits ist es möglich aus pcap-Paketen zu lesen. Das für diesen Zweck wichtigste Feature ist die Möglichkeit zum Verarbeiten eines Livestreams an Paketen wodurch es möglich ist die Pakete direkt vom Loopback Interface zu lesen. [42]
Damit von python auf die Datenbank zugegriffen werden kann muss ein Connector installiert werden. Die zugehörige Konfiguration findet sich wiederum in Abschnitt 9.4.

In weiterer Folge wurde, um die Daten persistent speichern zu können, ein MariaDB-Server installiert. Um unautorisierte root Logins zu vermeiden, wurde das Feature Mysql-Secure-Installation verwendet. Sämtliche sicherheitsrelevanten Einstellungen können Abschnitt 9.5 entnommen werden.
Weiters wurde im Code eine Funktion zum automatisierten Erstellen der gesamten Datenbankstruktur unter Verwendung von Default Werten entwickelt. Somit können alle Datenbanktabellen bei Bedarf auf deren ursprünglichen Zustand zurückgesetzt werden. Beim Zugriff auf die Datenbank wurde darauf geachtet, dass Prepared Statements verwendet werden um nicht durch das NIDS selbst ein Sicherheitsrisiko zu generieren.

Für das Versenden der Benachrichtigungen wird ein Postfix Mailserver mit gmail als Relayhost verwendet. Hierbei wurde ein GMAIL-Konto angelegt, über welches die E-Mails versendet werden. Es kann hierfür auch eine andere, bereits bestehende Mailadresse verwendet werden. Die Mailadresse inklusive Passwort muss lediglich im Konfigurationsfile ausgetauscht werden.

Diverse Einstellungen wurden in ein Konfigurationsfile ausgelagert. Dieses wird beim Starten des Programmes ausgelesen und beinhaltet datenbankspezifische Parameter sowie die Zugangsdaten zum Mailaccount, welcher für Benachrichtigungen verwendet wird. Im folgenden Snippet ist der Inhalt dieses Files ersichtlich:

```
pi@raspberrypi:~/sourcen/ZigBeeNIDS/ZigBeeNIDS $ cat config.ini
    [mysqlDB]
    host = localhost
    db = dbZigbeeNIDS
    user = ads_user
    pass = <your_password>
```

```
[mailSettings]
user = <your_mail>
pass = <your_password>
```

In Abschnitt 9.6 werden detaillierte Anweisungen gegeben, wie die Installation getestet werden kann. Dies soll vor allem die initiale Hürde zur Verwendung dieses Systems so gering als möglich gestalten.

6 Funktionstest des Network Intrusion Detection Systems

Um die Funktionalität des entwickelten Systems testen zu können wurde entsprechende Hardware angeschafft, um Angriffe auf ZigBee-Netzwerke durchführen zu können. Die Wahl fiel auf den Hersteller Attify[1], da sich dieser auf IoT Exploitation spezialisiert hat. Des Weiteren unterstützt dieser das Framework Killerbee, welches in weiterer Folge zur Durchführung der Angriffe verwendet wurde. Hardwaremäßig fiel die Wahl auf Grund des Preis-Leistungsverhältnisses auf ein Attify ApiMote v4 Beta[2].
Die zweite Form der Tests beruht auf aufgezeichneten Angriffen im pcap Format. So wurden auch Angriffe simuliert, um die Funktionalität des NIDS zu testen. Besonders bedanken möchte ich mich hiermit bei Tobias Zillner, BSc MSc MSc[3], welcher mir die pcap-Files zur Verfügung gestellt hat. Diese beinhalten Angriffe, die er im Zuge seiner Arbeit "ZigBee exploited: The good, the bad and the ugly" [21] veröffentlicht hat.

Der einzige Unterschied zwischen den Testmethoden besteht im Input des Systems. Hierbei gilt es zwischen Live-Traffic und zuvor aufgezeichneten Paketen, welche aus einem pcap-File gelesen werden, zu unterscheiden. Der Prozess des NIDS ist in beiden Fällen identisch. Zur Überprüfung der Funktionalität wird dieses Kapitel ebenso in die bereits definierten Angriffstypen unterteilt.

- Insecure Rejoin
- Replay
- Übernahme von Geräten
- Netzwerk-Key Sniffing

[1] https://www.attify.com/
[2] https://www.attify-store.com/products/apimote
[3] https://www.zillner.tech/

6.1 Insecure Rejoin

In den zur Verfügung gestellten pcap-Files ist unter anderem ein Insecure Rejoin enthalten. Um diesen zu erkennen, wird als Input das vorhandene pcap-File festgelegt und das NIDS gestartet. Dabei werden sämtliche ZigBee-Pakete ausgelesen, normalisiert und analysiert. Der Abbildung 6.1 können der Rejoin Request sowie der zugehörige Response im Wireshark entnommen werden. Hierbei sei anzumerken, dass das Security Flag auf false gesetzt ist. Dies bedeutet, dass keine Verschlüsselung der Kommunikation stattfindet und der Beitritt zum Netzwerk ein Sicherheitsrisiko darstellt.

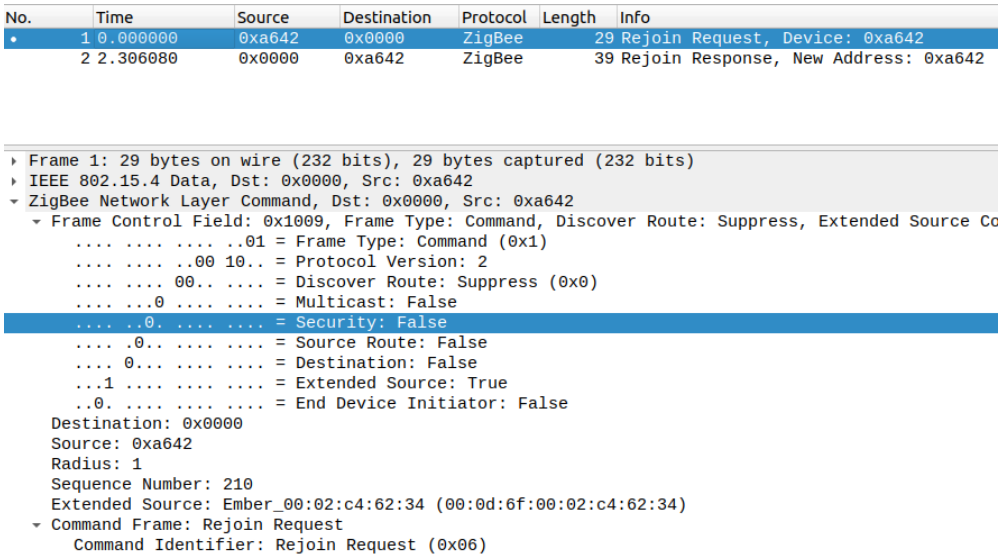

Abbildung 6.1: Insecure Rejoin - Wireshark

Nachdem das NIDS den Insecure Rejoin Response erkannt hat wird die Benachrichtigung, welche in Abbildung 6.2 ersichtlich ist, versendet.

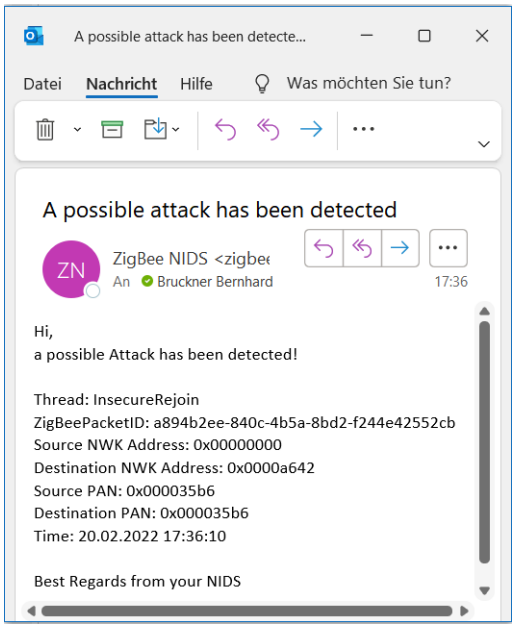

Abbildung 6.2: Insecure Rejoin - Benachrichtigung

6.2 Replay

Um einen Replay Angriff durchzuführen wird im ersten Schritt ein Angriffs-Setup, wie in Unterabschnitt 6.5.1 beschrieben, vorbereitet. In weiterer Folge wird das Framework Killerbee verwendet, um Daten aufzuzeichnen und diese anschließend erneut zu übertragen. Für diesen Test wurde der ”Lampe Ein” Befehl in einer Philips Hue Umgebung aufgezeichnet. Der letzten Spalte des Datenbankauszugs in Abbildung 6.3 kann entnommen werden, dass beim ersten Auftreten des Befehls keine Gefahr beziehungsweise Angriff erkannt wurde. Dieser Eintrag stellt den originalen Befehl dar. Nachdem der Request auf dem Testing System in einer Datei zwischengespeichert wurde kann dieser ebenso mittels Killerbee erneut an das Ziel gesendet werden. In weiterer Folge empfängt das NIDS exakt das gleiche Pakete erneut und führt dieses dem Analyseprozess zu. Daraus resultiert der zweite Eintrag in der Tabelle, welcher als Angriff erkannt wurde.

sourceaddress	destaddress	frame_type	cmd_id	security	zbee_sec_mic	zbee_sec_counter	nwk_seqno	isPossibleThread
0x00000001	0x0000fffd	0x00000000		1	a5becdfa	12393985	170	0
0x00000001	0x0000fffd	0x00000000		1	a5becdfa	12393985	170	1

Abbildung 6.3: Replay - Datenbankeintrag

Wie in Abbildung 6.4 ersichtlich können Details zum erkannten Angriff abgerufen werden. In dieser werden sämtliche Pakete aufgelistet, welche als Angriff erkannt wurden. Aus Gründen der Übersichtlichkeit

wurde die Datenbank regelmäßig geleert, wodurch zu diesem Zeitpunkt lediglich ein Treffer darin enthalten war. Dazu kommen die Namensinformationen des Angriffstyps aus der Tabelle tbThreadSettings, wobei es sich in diesem Fall um einen Replay handelt.

```
mysql> select uuid, sourceaddress, destaddress, zbee_sec_mic, zbee_sec_counter, nwk_seqno, ispossibleThread, tbThreadSettings.name, creationTime
    -> FROM tbZigBeeData
    -> LEFT JOIN tbDataThreadMapping ON tbZigBeeDataUUID=uuid
    -> LEFT JOIN tbThreadSettings ON tbThreadSettings.id = tbThreadSettingsID
    -> where isPossibleThread = 1;
+--------------------------------------+---------------+-------------+-------------+-----------------+----------+-----------------+--------+---------------------+
| uuid                                 | sourceaddress | destaddress | zbee_sec_mic | zbee_sec_counter | nwk_seqno | ispossibleThread | name   | creationTime        |
+--------------------------------------+---------------+-------------+-------------+-----------------+----------+-----------------+--------+---------------------+
| f52cf3cc-e110-459e-85da-21e5c2bec5b9 | 0x00000001    | 0x0000fffd  | a5becdfa    | 12393985        | 170      |               1 | Replay | 2022-03-04 19:18:17 |
+--------------------------------------+---------------+-------------+-------------+-----------------+----------+-----------------+--------+---------------------+
1 row in set (0.01 sec)
```

Abbildung 6.4: Replay - Datenbankeintrag inklusive Details

Um das Bild des Angriffs zu vervollständigen findet sich in Abbildung 6.5 die zugehörige Benachrichtigung.

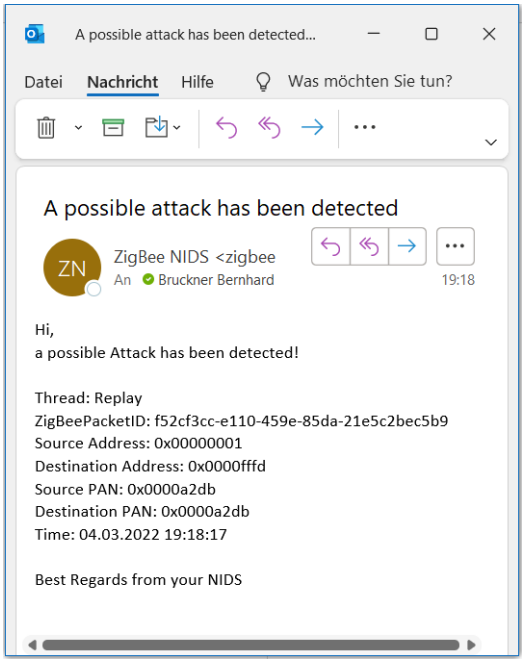

Abbildung 6.5: Replay - Benachrichtigung

6.2.1 Verschlüsselter versus unverschlüsselter Traffic

Es wurde beim Design des NIDS darauf geachtet, dass Replay Angriffe in beiden Fällen erkannt werden. Es ist folglich für die Funktionalität des Systems irrelevant, ob die Angreifer*innen verschlüsselte oder unverschlüsselte Befehle erneut senden. Um dies zu erklären ist in Abbildung 6.6 der Befehl ”Lampe Ein” des zuvor durchgeführten Replay Angriffs in unverschlüsselter Form abgebildet. Als Spalten wurden der verwendete Frame Counter, der Message Integrity Code sowie die Sequence Number hinzugefügt. Wie bereits erwähnt dienen diese Felder als Hauptindikatoren für das NIDS um Replay Angrif-

fe zu erkennen. So ist es auch möglich Angreifer*innen, welche bereits im Besitz des Netzwerk-Keys sind und einen Replay Angriff durchführen zu erkennen. Ein weiterer Vorteil ist, dass dazu der eigene Netzwerk-Key nicht im NIDS hinterlegt werden muss.

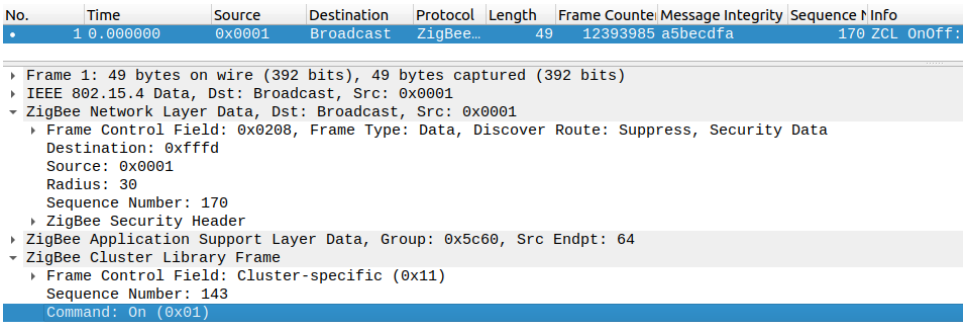

Abbildung 6.6: Replay ”Lampe Ein” - Wireshark

In Abbildung 6.7 ist der gleiche Befehl erneut, aber in verschlüsselter Form, abgebildet. Dieser kann entnommen werden, dass der Frame Counter, der Message Integrity Code sowie die Sequence Number sowohl aus dem unverschlüsseltem als auch aus dem verschlüsselten Paket mit dem zuvor aufgezeichneten übereinstimmen. Zusammengefasst kann man sagen, dass das NIDS Replay Angriffe erkennen kann, unabhängig davon ob die Pakete verschlüsselt oder unverschlüsselt übertragen werden.

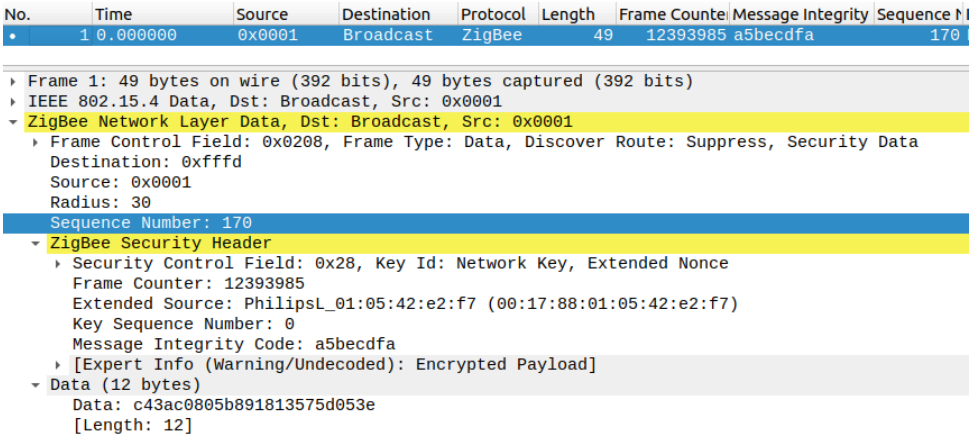

Abbildung 6.7: Replay ”Lampe Ein” (verschlüsselt) - Wireshark

Die für die Ausführung notwendigen Befehle finden sich im Anhang unter Abschnitt 9.7.

Ebenso werden die pcap-Files auf Replay Angriffe untersucht. Dabei wird festgestellt, dass darin Pakete in anormaler Form mehrfach vorkommen. In Abbildung 6.8 ist die daraus resultierende Benachrichtigung des NIDS ersichtlich.

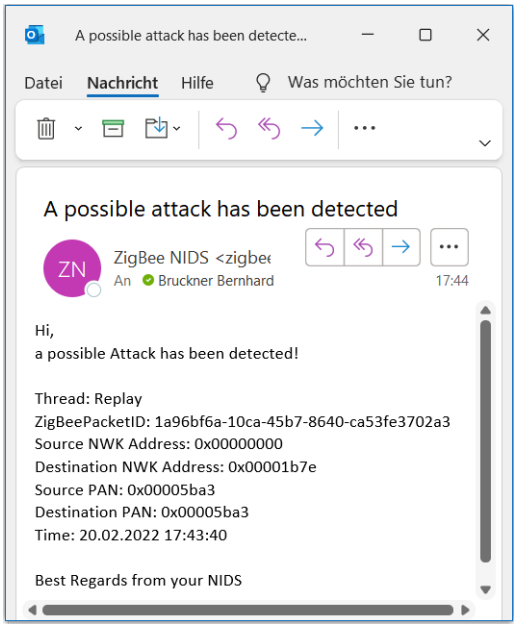

Abbildung 6.8: Replay - Benachrichtigung

6.3 Übernahme von Geräten

Bei der Übernahme von Geräten werden vom NIDS zwei verschiedene Muster überprüft. Zum einen wird auf den ”ResetToFactory” Befehl geprüft, da das Verlassen des Netzwerks Grundvoraussetzung für die Übernahme eines ZigBee-Geräts ist. Des Weiteren werden vom NIDS alle Network Join Requests analysiert, wobei sowohl auf ”Network Join Router Request” als auch ”Network Join End Device Request” überprüft wird. Wird einer dieser beiden Requests erkannt wird dessen PAN-ID in weiterer Folge mit der von dem/der Benutzer*in hinterlegten PAN-ID verglichen. Dieser Angriff wurde nicht praktisch durchgeführt. Das Wissen über die Funktionalität und die entsprechenden Felder im ZigBee Frame wurde mittels Reverse Engineering des Tools Z3sec[4] erlangt. Des Weiteren wurden die zugehörigen Command Identifier dem ZigBee Light Link User Guide [43] entnommen.

Um das Erkennen des ersten Musters zu überprüfen wurde in der Testumgebung eine Philips Hue Bridge, ein Philips Hue Dimmer Switch und einige Philips Hue Glühbirnen und Lampen verwendet. Den Normalbetrieb stellt hierbei die Steuerung der Glühbirne mittels Dimmer Switch dar. Mittels Attify ApiMote, aber auch mit dem mitgelieferten Philips Hue Dimmer Switch ist es möglich Factory Reset Befehle an eine Lampe zu senden. Beim Dimmer Switch von Philips muss dieser dabei sehr nahe an der Philips Glühbirne positioniert werden. Danach kann mittels Tastenkombination die Lampe dazu veranlasst werden sich auf Werkseinstellungen zurückzusetzen. Dabei blinkt die Lampe ein paar Mal. Dieser Vor-

[4] https://github.com/IoTsec/Z3sec/blob/master/z3sec/

gang wurde zum Test der Installation durchgeführt. Kurz nach dem Zurücksetzen der Lampe wurde eine Benachrichtigung vom NIDS erhalten, dass ein FactoryReset Befehl erkannt wurde. Wie man Abbildung 6.10 entnehmen kann verfügt ein FactoryToReset Befehl über keine NWK Source- und Destination Adresse, wodurch die Adressen aus dem WPAN Header verwendet werden. Die darin ersichtlichen zusätzlichen Header (Ethernet II, Internet Protocol Version 4, User Datagram Protocol und ZigBee Encapsulation Protocol) entstehen durch die Verwendung des ConBee II und können in weiterer Folge ignoriert werden.

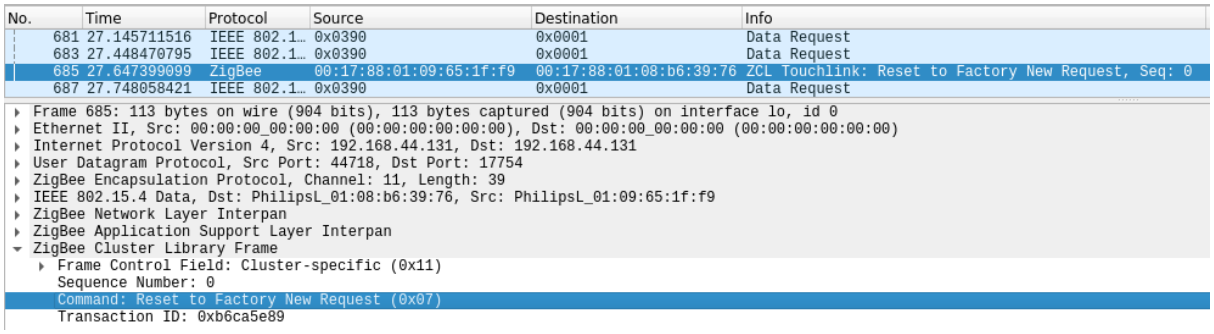

Abbildung 6.9: FactoryReset - Wireshark

In Abbildung 6.10 ist der detaillierte Auszug aus der Datenbank zu dem erkannten ResetToFactory Befehl ersichtlich.

```
mysql> select uuid, sourceaddress, destaddress,  nwk_seqno, ispossibleThread, tbThreadSettings.name, creationTime from tbZigBeeData
    -> LEFT JOIN tbDataThreadMapping ON tbZigBeeDataUUID=uuid
    -> LEFT JOIN tbThreadSettings ON tbThreadSettings.id = tbThreadSettingsID
    -> where isPossibleThread = 1;
+--------------------------------------+-----------------------+-----------------------+-----------+-----------------+---------------+---------------------+
| uuid                                 | sourceaddress         | destaddress           | nwk_seqno | ispossibleThread | name          | creationTime        |
+--------------------------------------+-----------------------+-----------------------+-----------+-----------------+---------------+---------------------+
| 34c55663-7b05-4fad-9a23-3f3102f54ccb | 00:17:88:01:09:65:1f:f9 | 00:17:88:01:08:b6:39:76 | 241       |               1 | ResetToFactory | 2022-02-20 13:19:11 |
+--------------------------------------+-----------------------+-----------------------+-----------+-----------------+---------------+---------------------+
1 row in set (0.00 sec)
```

Abbildung 6.10: FactoryReset - Datenbank

Der Vollständigkeitshalber wird in Abbildung 6.11 die zugehörige Benachrichtigung dargestellt.

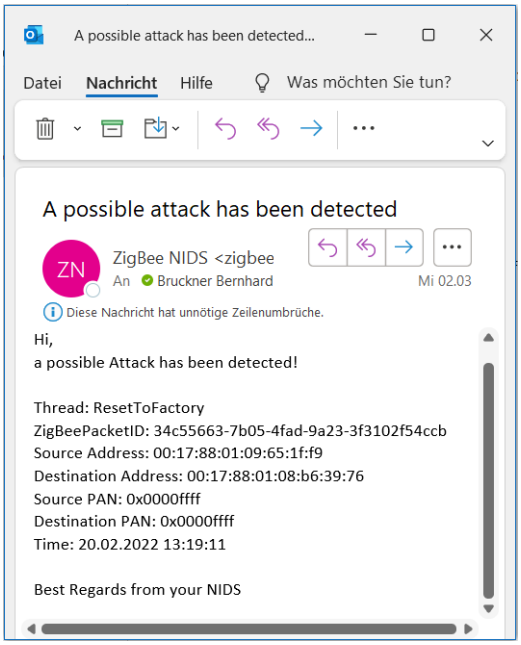

Abbildung 6.11: FactoryReset - Benachrichtigung

6.4 Netzwerk-Key Sniffing

Um Überprüfen zu können ob das Übertragen des Netzwerk-Keys erkannt wird wurde im ersten Schritt die Philips Lampe auf Werkseinstellungen zurückgesetzt. Im nächsten Schritt wurde diese mittels Light Link Commissioning dem Netzwerk hinzugefügt. Dabei wird der Netzwerk-Key, mittels Light Link Commissioning Key verschlüsselt, an die Lampe übertragen. Da der öffentlich bekannte LLCK im Wireshark hinterlegt ist wird dieser Vorgang automatisiert entschlüsselt und das System erkennt die Übertragung des Netzwerk-Keys. In Abbildung 6.12 ist der zugehörige Wireshark-Ausschnitt ersichtlich. Diesem kann entnommen werden, dass die Übertragung des Netzwerk-Keys mittels Light Link Commissioning Key geschützt war. Da dieser aber öffentlich bekannt ist und somit als unsicher gilt wird eine Gefahr erkannt.

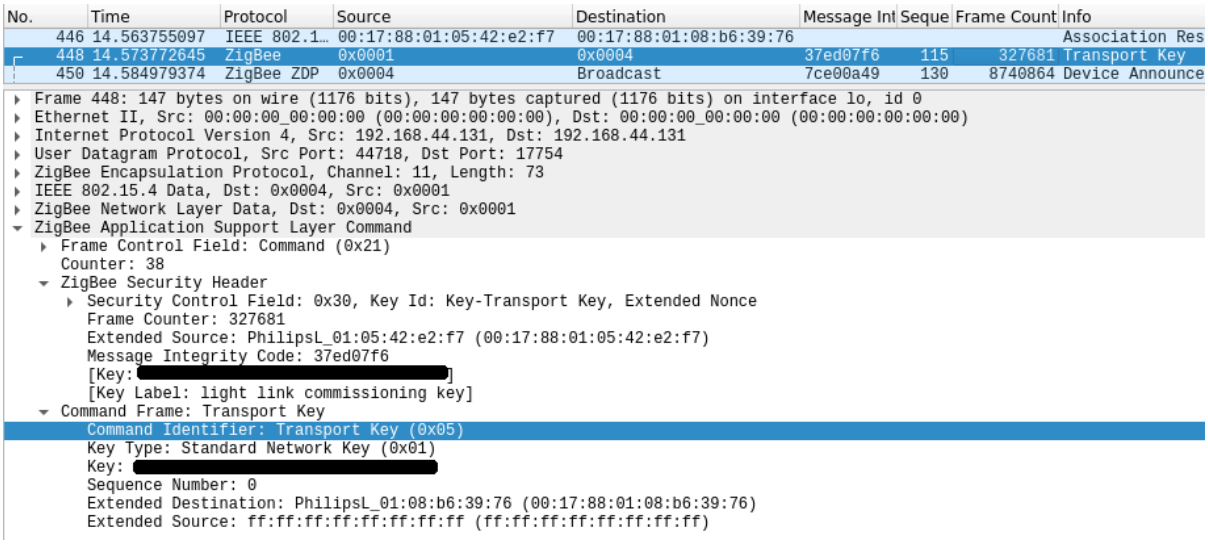

Abbildung 6.12: Transport-Key - Wireshark

In Abbildung 6.13 finden sich zugehörige Details aus der Datenbank des NIDS. Dieser kann entnommen werden, dass ein ”TransportKeyCommand” erkannt wurde. Des Weiteren entsprechen die Source- und Zieladresse den in Abbildung 6.12 ersichtlichen.

```
mysql> select uuid, sourceaddress, destaddress, zbee_sec_mic, nwk_seqno, ispossibleThread, tbThreadSettings.name, creationTime from tbZigBeeData
    -> LEFT JOIN tbDataThreadMapping ON tbZigBeeDataUUID=uuid
    -> LEFT JOIN tbThreadSettings ON tbThreadSettings.id = tbThreadSettingsID
    -> where uuid = 'c9c2236e-048a-4a62-8ee7-d86832015529';
+--------------------------------------+---------------+-------------+-------------+-----------+-----------------+---------------------+---------------------+
| uuid                                 | sourceaddress | destaddress | zbee_sec_mic | nwk_seqno | ispossibleThread | name                | creationTime        |
+--------------------------------------+---------------+-------------+-------------+-----------+-----------------+---------------------+---------------------+
| c9c2236e-048a-4a62-8ee7-d86832015529 | 0x00000001    | 0x00000004  |             | 115       |               1 | TransportKeyCommand | 2022-03-02 17:29:13 |
+--------------------------------------+---------------+-------------+-------------+-----------+-----------------+---------------------+---------------------+
1 row in set (0.11 sec)
```

Abbildung 6.13: Transport-Key - Datenbank

Der Vollständigkeitshalber findet sich in Abbildung 6.14 wiederum die zugehörige Benachrichtigung. Dieser kann ebenso entnommen werden, dass es sich um ein ”TransportKeyCommand” handelt. Dies bedeutet für die Besitzer*innen, dass der Netzwerk-Key ungeschützt oder nur mit Default-Key verschlüsselt übertragen wurde.

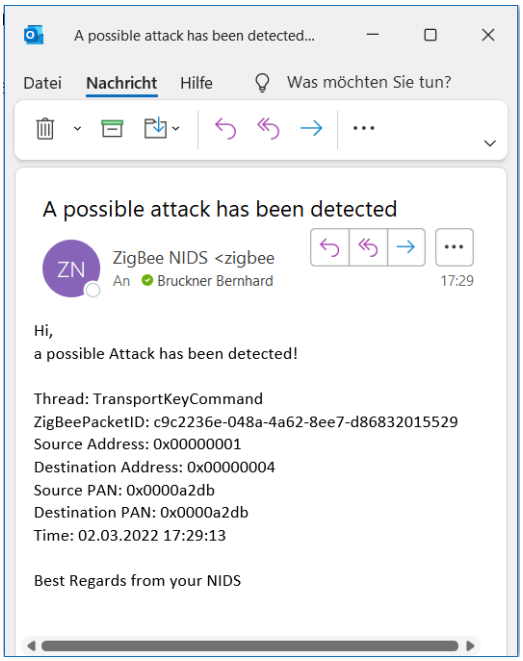

Abbildung 6.14: Transport-Key - Benachrichtigung

Ebenso wurden die pcap-Files dahingehend analysiert, ob während der Aufzeichnung der Netzwerk-Key ungeschützt übertragen wurde. Dazu zählt die unverschlüsselte sowie die nur mit einem Default Key verschlüsselte Übertragung. Beim Durchlauf der pcap-Files durch den Analyse Prozess des NIDS wurde ein ungeschütztes Transport-Key Kommando erkannt und eine entsprechende Benachrichtigung versendet. Der zugehörige Auszug aus dem Wireshark findet sich in Abbildung 6.15. Diesem kann entnommen werden, dass die Übertragung des Netzwerk-Keys mit dem Default Trust Center Link Key verschlüsselt wurde.

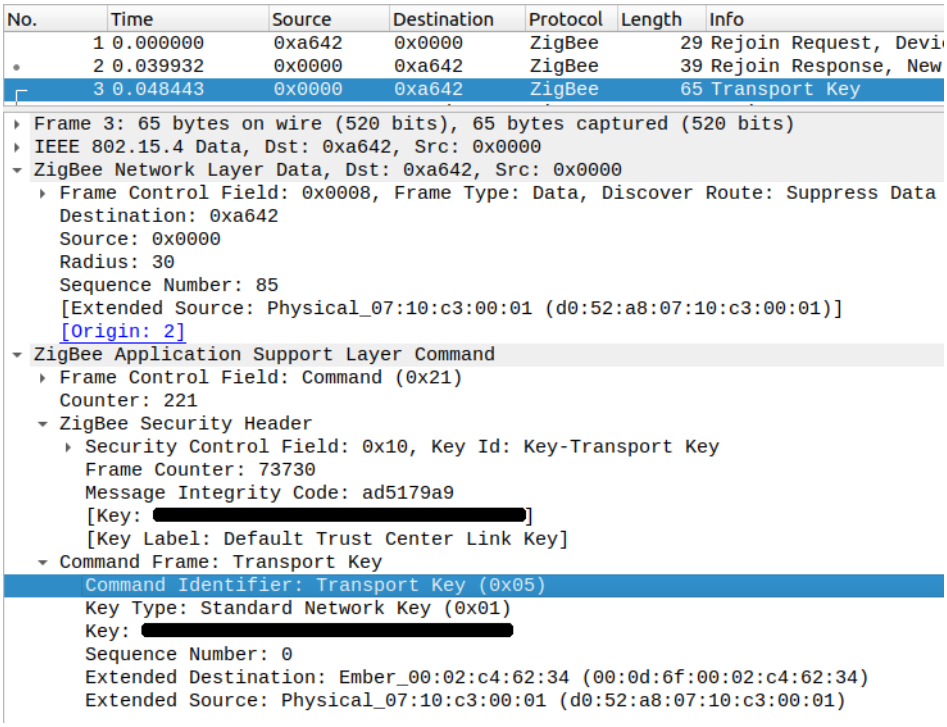

Abbildung 6.15: Netzwerk-Key Sniffing - Wireshark

Die generierte Benachrichtigung ist in Abbildung 6.16 ersichtlich.

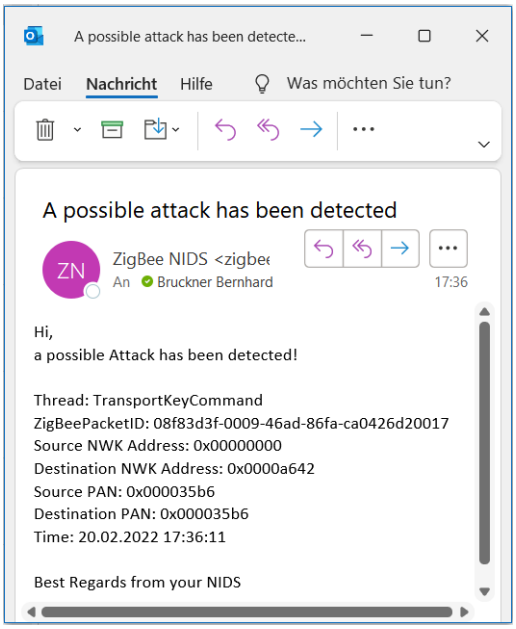

Abbildung 6.16: Netzwerk-Key Sniffing - Benachrichtigung

6.5 Test des Systems auf false-positives

Im Gegensatz zu den vorherigen Tests wurde auch die Anfälligkeit auf false-positives getestet. Dazu wird das funktionierende Netzwerk temporär deaktiviert, wodurch der Philips Dimmer Switch die Verbindung zu seinem Parent verliert und in weiterer Folge dazu bewegt wird einen Rejoin durchzuführen. Auf Grund der zuvor funktionierenden Installation ist dieser Rejoin jedoch mit dem aktuellen Netzwerk-Key gesichert. Wie man der Abbildung 6.17 entnehmen kann wird der gesendete Befehl gleich vier Mal vom ConBee II aufgezeichnet. Es kann sichergestellt werden, dass der Hue Dimmer Switch den Befehl nicht selbst vier Mal gesendet hat, da ansonsten die Sequence Number und der Frame Counter erhöht worden wären. Des Weiteren wurde parallel dazu mit dem ApiMote v4 derselbe Traffic aufgezeichnet, wobei in diesem Fall der Rejoin Request Befehl nur einmal im Wireshark angekommen ist. Deshalb ist davon auszugehen, dass es sich hierbei um ein Problem seitens des ConBee II handelt. Es sei angemerkt, dass dieses Phänomen lediglich beim Rejoin Request auftritt. Die restliche Kommunikation wird auch vom ConBee II korrekt aufgezeichnet. Nichtsdestotrotz wird aber kein Insecure Rejoin erkannt, da hierbei das Security Flag auf ”1” gesetzt ist und somit die Kommunikation verschlüsselt stattfindet. Dieser Wert ist zusätzlich in der Abbildung hervorgehoben. Der Rejoin wird also nicht unverschlüsselt übertragen, sondern wie bereits erwähnt durch den Netzwerk-Key geschützt. In diesem Fall kann das NIDS beziehungsweise Wireshark den Traffic ebenso interpretieren, da der Netzwerk-Key für diesen Test im Wireshark hinterlegt wurde. Zusammengefasst hat also ein Netzwerk Rejoin eines Gerätes innerhalb des ZigBee-Netzwerks stattgefunden, wobei richtigerweise kein Angriff erkannt wurde. Dies stellt den normalen und sicheren Ablauf dar, falls ein Gerät die Verbindung zu seinem Parent verliert.

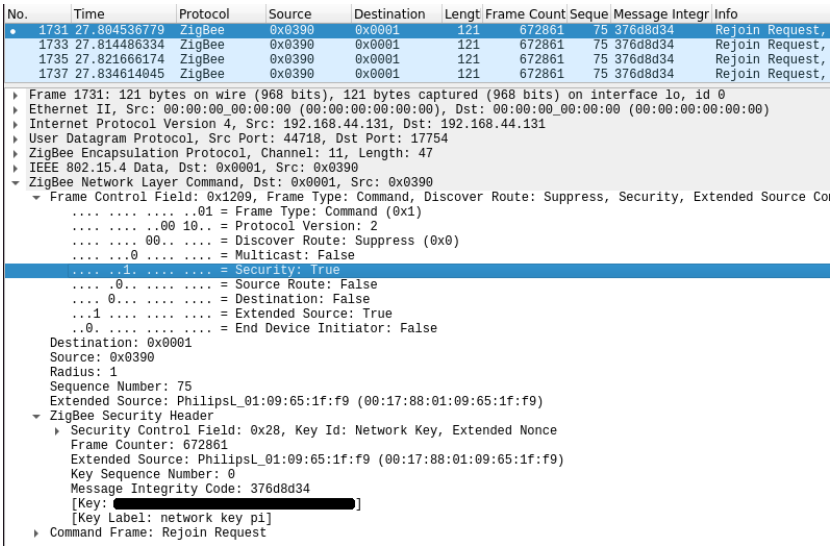

Abbildung 6.17: Test false-positive Wireshark

Da exakt das gleiche Paket mehrmals aufgezeichnet wird erkennt das NIDS einen Replay Angriff. Um dies besser visualisieren zu können wird in Abbildung 6.18 noch der zugehörige Auszug aus der Datenbank dargestellt, wobei aus Gründen der besseren Lesbarkeit die Spalten namens id, uuid und creationTime nicht am Screenshot enthalten sind. Diesem kann man entnehmen, dass es sich um exakt denselben Befehl handelt. In der letzten Spalte wird festgehalten, dass in diesem Fall beim ersten Paket keine Gefahr erkannt wurde und es sich ab dem zweiten Paket um einen potenziellen Angriff handelt. Dies ist dem Umstand geschuldet, dass exakt dasselbe Paket mehrfach aufgezeichnet wird. Die zugehörige Benachrichtigung findet sich in Abbildung 6.19.

sourceaddress	destaddress	frame_type	cmd_id	security	zbee_sec_mic	zbee_sec_counter	nwk_seqno	isPossibleThread
0x00000390	0x00000001	0x00000001	0x00000006	1	376d8d34	672861	75	0
0x00000390	0x00000001	0x00000001	0x00000006	1	376d8d34	672861	75	1
0x00000390	0x00000001	0x00000001	0x00000006	1	376d8d34	672861	75	1
0x00000390	0x00000001	0x00000001	0x00000006	1	376d8d34	672861	75	1

Abbildung 6.18: Test false-positive Datenbank

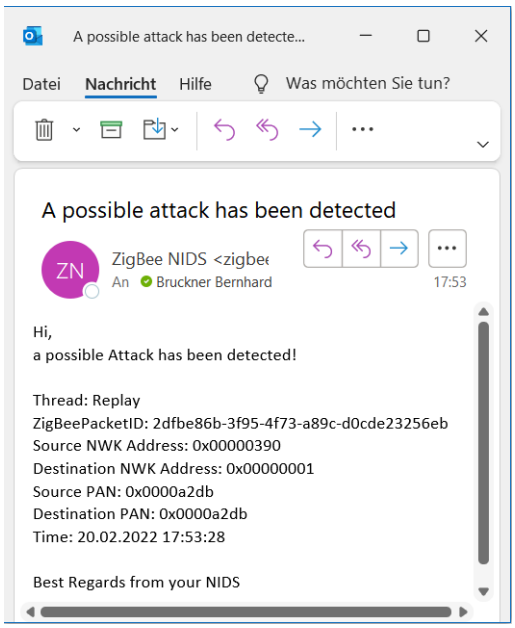

Abbildung 6.19: Replay Test via Rejoin Request - Benachrichtigung

6.5.1 Testing System

Um die oben genannten praktischen Angriffe durchführen zu können wurde ein System aufgebaut, mit dem es möglich ist Angriffe auf ZigBee Netzwerke durchzuführen. Dieses besteht aus einem Ubuntu 20.04 LTS, auf welchem diverse Frameworks installiert werden. Als Programmiersprache wird, wie beim NIDS, python verwendet. Als Hardware dient ein Attify ApiMote v4 Beta.

Im ersten Schritt wird Scapy installiert. Scapy ist eine Bibliothek beziehungsweise ein Programm, welches auf Python basiert und zur interaktiven Manipulation von Paketen verwendet werden kann. Mittels scapy ist es also möglich Pakete zu erzeugen, zu empfangen und zu dekodieren. [44] Im nächsten Schritt wird das Killerbee Framework installiert, wobei die Details zur Installation Abschnitt 9.7 entnommen werden können. In weiterer Folge kann mit dem Befehl zbid kontrolliert werden, ob das per USB verbundene ApiMote erkannt wird. Hierbei sollte der tty aufscheinen, den das ApiMote belegt.
Die Tests wurden mit dem Framework Killerbee[5] durchgeführt. Bei diesem handelt es sich um ein Framework, welches ebenso auf python basiert. Dabei ist es möglich ZigBee Traffic aufzuzeichnen, zu manipulieren, abzufangen und zu dekodieren. Auf Grund der geringen Abhängigkeiten ist dieses einfach zu verwenden und erweiterbar. [26]

[5] https://github.com/riverloopsec/killerbee

7 Sicherer Betrieb eines ZigBee-Netzwerks

Um die Frage, ob man ein ZigBee-Netzwerk sicher genug betreiben kann, beantworten zu können muss sich, wie so oft in der IT-Sicherheit, die Frage nach dem Risiko gestellt werden. Es ist für eine Privatperson auf Grund der notwendigen geografischen Nähe sehr unwahrscheinlich, dass sich diese mit einem Angriff auf ihr ZigBee-Netzwerk auseinandersetzen muss. Hierbei ist die Gefahr durch einen schadhaften Link in einer E-Mail weitaus größer. Meiner Meinung nach spricht also nichts dagegen als Privatperson ZigBee-fähige Produkte zu verwenden, wobei im besten Fall zusätzlich das NIDS integriert wird. Als weitere Sicherheitsmaßnahme könnte man zwischen LAN und dem Trust Center eine Firewall verwenden, wobei der Traffic lediglich Richtung Internet erlaubt ist. So kann sichergestellt werden, dass aus dem ZigBee Netzwerk nicht auf Daten im LAN zugegriffen werden kann. Da ohnehin keine großen Datenmengen aus dem ZigBee-Netzwerk zu erwarten sind kann hierbei beispielsweise eine pfsense[1] verwendet werden. Dies ist eine weit verbreitete Open Source Firewall, welche auf dem Betriebssystem FreeBSD basiert.

Bei großen Firmen oder international tätigen Konzernen wird das Ergebnis der Risikoanalyse ein anderes sein. Diese verfügen meist über eine gut abgesicherte IT-Infrastruktur wobei oft 802.1x verwendet wird, um sich vor unautorisierten Geräten im Netzwerk zu schützen. Wird nun dieses Netzwerk um ZigBee-fähige Geräte erweitert muss auf diese Erweiterung besonderes Augenmerk gelegt werden, um keine Schwachstelle zu generieren. Des Weiteren stellen diese auch für Angreifer*innen ein weitaus lohnenderes Ziel dar.

[1] https://www.pfsense.org/

8 Conclusion

In diesem Kapitel werden die Forschungsfragen mit Hilfe der gewonnenen Erkenntnisse beantwortet. Ebenso werden Möglichkeiten für künftige Arbeiten, welche auf dem NIDS aufbauen, aufgezeigt.
Die immer größere Verbreitung von IoT steigert auch die Beliebtheit von einfach zu verwendenden Funkprotokollen wie ZigBee. So gibt es immer mehr Anwendungsfälle und die Anzahl der ZigBee-Geräte steigt kontinuierlich. Leider steigert dies auch das Interesse von potenziellen Angreifern und Angreiferinnen um sich damit zu beschäftigen. Auf Grund der vielfältigen Angriffsmöglichkeiten auf ZigBee wurde eine Lösung entwickelt um den Beitrieb eines solchen Netzwerks sicherer zu machen.
Die in Abschnitt 1.4 definierten Forschungsfragen lauten wie folgt:

- Ist es generell möglich Muster in Angriffen auf ZigBee-Netzwerke zu erkennen?
- Wie kann man Angriffe klassifizieren, automatisiert erkennen und die Besitzer*innen im Bedarfsfall benachrichtigen?

Nachdem verschiedenste Angriffsszenarien aus der Literatur analysiert waren, wurden die gängigsten davon in vier Angriffstypen unterteilt. Die Klassifizierung dieser Angriffstypen ermöglicht es bestimmte Muster abzuleiten und in einer Datenbank zu hinterlegen. Dabei unterscheiden sich die Angriffe im Format der Frames, der Häufigkeit der Übermittlung sowie bestimmten Abläufen. Mittels Raspberry PI und ConBee II wird der aktuelle ZigBee-Traffic aufgezeichnet, normalisiert und gegen die Angriffsmuster in der Datenbank geprüft. Stimmt das aktuelle ZigBee-Paket mit einem der Muster überein so wird, je nach Konfiguration, eine Benachrichtigung an die Besitzer*innen versendet. Somit können Angriffe automatisiert erkannt, sowie in weiterer Folge die Benutzer*innen darüber informiert werden.
Um das NIDS verwenden zu können benötigt man eine Möglichkeit um ZigBee-Traffic aufzeichnen zu können. Im Optimalfall handelt es sich hierbei um einen ConBee II beziehungsweise RaspBee II. Derzeit wird nur ein ZigBee-Kanal aufgezeichnet. Dies kann jedoch einfach durch Verwendung mehrerer Dongles geändert werden.

8.1 Weiterführende Arbeiten

In einer weiterführenden Arbeit könnte das entwickelte NIDS um mehrere ConBee II ergänzt werden, um auch weitere Kanäle abdecken zu können. Dabei müsste an der Konfiguration des NIDS nichts geändert werden, sondern lediglich der Traffic von zshark wiederum an das Loopback Interface gesendet werden. Eine weitere spannende Weiterentwicklung wäre, aus dem NIDS ein Container-Image zu erstellen. So wäre die Installation noch einfacher, da keine Abhängigkeiten mehr installiert werden müssten. Dabei könnten das Konfigurationsfile und einige wenige Parameter dem Container beim Startup übergeben werden, wodurch die Konfiguration persistent gespeichert werden kann.
Abschließend könnte man die bereits zuvor erwähnte Sicherheitsmaßnahme zur Trennung von LAN und ZigBee-Netzwerk mittels Firewall implementieren.

9 Anhang

9.1 Raspberry PI Konfiguration

```
# sudo apt-get install python3 python3-pip
# sudo apt-get install libqt5serialport5
# sudo dpkg -i zshark-1.00.05.deb
# sudo apt-get install -f
```

9.2 Zshark Installation

```
# wget https://phoscon.de/downloads/zshark/raspbian/zshark-1.00.05.deb
# sudo dpkg -i zshark-1.00.05.deb
# sudo apt-get install libqt5serialport5
```

9.3 Wireshark Installation

```
# sudo apt-get install wireshark
# sudo usermod -a -G wireshark $USER
# sudo chmod +x /usr/bin/dumpcap
```

9.4 Pyshark Installation

```
# pip3 install pyshark==0.4.3
# sudo apt-get install -y tshark
# pip3 install mysql-connector-python-rf
```

9.5 MariaDB-Server Installation

```
# sudo apt-get install mariadb-server
```

```
# sudo mysql_secure_installation

Set root password? [Y/n] y
Remove anonymous users? [Y/n] y
Disallow root login remotely? [Y/n] y
Remove test database and access to it? [Y/n] y
Reload privilege tables now? [Y/n] y
```

Mit folgendem Befehl kann überprüft werden ob der mysql-Dienst auf Port 3306 lauscht.

```
# ss -ltn
```

Um nun eine Verbindung zum lokalen Mysql-Server aufzubauen kann der folgende Befehl verwendet werden.

```
# sudo mysql -u root -p
```

Eine Datenbank inklusive Benutzer mit entsprechenden Berechtigungen kann wie im folgenden Snippet ersichtlich erstellt werden.

```
# MariaDB [(none)]> create database dbZigBeeNIDS;
# MariaDB [(none)]> CREATE USER 'nids_user'@'%'
                    IDENTIFIED BY '<your_password>';
# MariaDB [(none)]> GRANT ALL privileges on dbZigBeeNIDS.*
                    to nids_user@'%';
# MariaDB [(none)]> flush privileges;
```

9.6 Testen der Installation

9.6.1 Datenbankverbindung prüfen

Um die Verbindung zur Datenbank zu überprüfen kann man sich mit dem folgenden Befehlen am lokalen MariaDB-Server anmelden. Des Weiteren kann getestet werden, ob der Benutzer ausreichend Berechtigungen hat um auf die angelegte Datenbank zuzugreifen.

```
# mysql -u nids_user -p
# MariaDB [(none)]> show databases;
# +--------------------+
# | Database           |
# +--------------------+
# | dbZigbeeNIDS       |
# | information_schema |
# +--------------------+
# 2 rows in set (0.001 sec)
```

9.6.2 Datenbankzugriff via python überprüfen

Um den Zugriff auf die Datenbank aus der python Umgebung zu testen, kann die Methode dbCon.testdbcon() verwendet werden. Folgend sieht man die relevanten Zeilen in einem Code-Snippet.

```
# dbCon = db.DatabaseConnection(...)
# dbCon.testdbcon()
# return
```

Das erfolgreiche Ergebnis dieses Tests sieht wie folgt aus:

```
# python3 ZigBeeNIDS.py
# Connected to MySQL Server version  5.5.5-10.3.31-MariaDB..
# You're connected to database:  ('dbZigbeeNIDS',)
# MySQL connection is closed
```

9.6.3 Automatisierte Erstellung der Datenbankstruktur

Mit der Funktion dbCon.recreateDatabaseStructure() ist es möglich die gesamte Datenbankstruktur automatisiert zu erstellen.

```
# python3 ZigBeeNIDS.py
# #####
# MariaDB [dbZigbeeNIDS]> show tables;
# +-----------------------+
# | Tables_in_dbZigBeeNIDS |
```

```
# +-----------------------+
# | tbDataThreadMapping   |
# | tbGeneralSettings     |
# | tbThreadSettings      |
# | tbZigBeeData          |
# +-----------------------+
# 4 rows in set (0.001 sec)
```

9.6.4 Auslesen aus pcap File

Sollte man aus einem pcap-File lesen wollen, ist es notwendig dessen Pfad in der FileCapture Methode anzugeben.

```
# capture = pyshark.FileCapture('/<your_path>/test_reset.pcap')
# for packet in capture:
```

9.6.5 Test Wireshark

Nachdem zshark, wie in Abschnitt 5.9 beschrieben, konfiguriert und gestartet wurde kann im nächsten Schritt Wireshark aufgerufen werden. Hierbei ist darauf zu achten, dass dieses mit root Rechten ausgeführt wird, damit das Aufzeichnen des Loopback Interfaces möglich ist. In Abbildung 9.1 ist die Funktionsweise inklusive der definierten Filter ersichtlich.

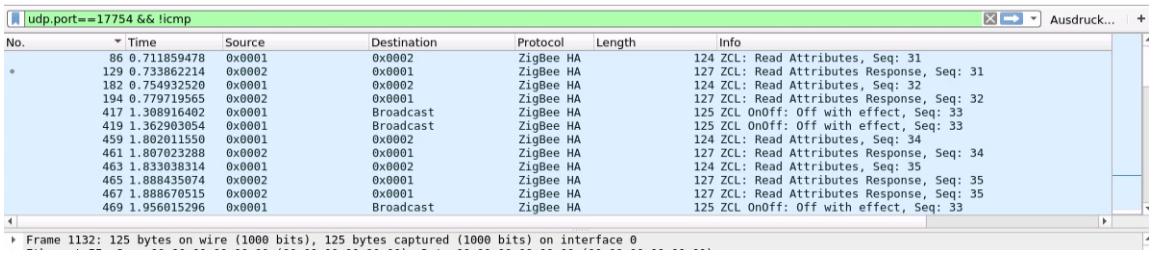

Abbildung 9.1: Wireshark Test

9.7 Installation und Verwendung von Killerbee

```
# sudo apt-get install python-gtk2 python-cairo python-usb
  python-crypto python-serial python-dev-is-python2 libgcrypt20-dev
# git clone https://github.com/secdev/scapy
```

```
# cd scapy
# sudo python3 setup.py install

# git clone https://github.com/riverloopsec/killerbee/tree/master
# cd killerbee-master
# sudo python3 setup.py install



1. Aufzeichnen des Lampe Ein Befehls und speicherns in ein pcap File
# zbdump -w replay_lampOn.pcap -c 11 -n 20

2. Replay der aufgezeichneten Pakete mittels killerbee
# zbreplay -c 11 -r replay_lampOn.pcap -n 5
```

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] S. Rana, M. A. Halim, and M. H. Kabir, "Design and implementation of a security improvement framework of zigbee network for intelligent monitoring in iot platform," *Applied Sciences*, vol. 8, no. 11, p. 2305, 2018.

[2] O. Olawumi, K. Haataja, M. Asikainen, N. Vidgren, and P. Toivanen, "Three practical attacks against zigbee security: Attack scenario definitions, practical experiments, countermeasures, and lessons learned," in *2014 14th International Conference on Hybrid Intelligent Systems*. IEEE, 2014, pp. 199–206.

[3] S. U. S. estimates, "Globale Absatzentwicklung smarter Außenbeleuchtung bis 2022," 2016, letzter Zugriff: 09. Oktober 2021. [Online]. Available: https://de.statista.com/statistik/daten/studie/685348/umfrage/absatzentwicklung-von-aussenbeleuchtung-nach-region-weltweit/

[4] S. Unlimited, "Globale Absatzentwicklung smarter Außenbeleuchtung bis 2022," 2016, letzter Zugriff: 09. Oktober 2021. [Online]. Available: https://de.statista.com/statistik/daten/studie/648296/umfrage/led-anteil-smarter-aussenbeleuchtung-weltweit/

[5] ——, "Globale Absatzentwicklung smarter Außenbeleuchtung bis 2022," 2016, letzter Zugriff: 09. Oktober 2021. [Online]. Available: https://de.statista.com/statistik/daten/studie/680003/umfrage/absatzanteil-von-led-aussenbeleuchtung-weltweit-nach-datenuebertragungssystem/

[6] G. Lee, J. Lim, D.-k. Kim, S. Yang, and M. Yoon, "An approach to mitigating sybil attack in wireless networks using zigbee," in *2008 10th International Conference on Advanced Communication Technology*, vol. 2. IEEE, 2008, pp. 1005–1009.

[7] D.-G. Akestoridis and P. Tague, "Hiveguard: A network security monitoring architecture for zigbee networks," in *2021 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2021, pp. 209–217.

[8] F. Sadikin and S. Kumar, "Zigbee iot intrusion detection system: A hybrid approach with rule-based and machine learning anomaly detection." in *IoTBDS*, 2020, pp. 57–68.

[9] Q. Sun, X. Miao, Z. Guan, J. Wang, and D. Gao, "Spoofing attack detection using machine learning in cross-technology communication," *Security and Communication Networks*, vol. 2021, 2021.

[10] S. C. Ergen, "Zigbee/ieee 802.15. 4 summary," *UC Berkeley, September*, vol. 10, no. 17, p. 11, 2004.

[11] H. HKCERT, "Iot device (zigbee) security study," *HKCERT Blog – https://www.hkcert.org/f/blog/264453/3a1c8eed-012c-4b59-9d9e-971001d66c77-DLFE-14602.pdf*, 04 2020.

[12] G. Dini and M. Tiloca, "Considerations on security in zigbee networks," in *2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*. IEEE, 2010, pp. 58–65.

[13] C. S. Chin, W. Atmodihardjo, W. L. Woo, and E. Mesbahi, "Remote temperature monitoring device using a multiple patients-coordinator set design approach," *ROBOMECH Journal*, vol. 2, 12 2015.

[14] M. Hillman, "An overview of zigbee networks," letzter Zugriff: 15.03.2022. [Online]. Available: https://www.f-secure.com/en/consulting/our-thinking/an-overview-of-zigbee-networks

[15] S. Khanji, F. Iqbal, and P. Hung, "Zigbee security vulnerabilities: Exploration and evaluating," in *2019 10th International Conference on Information and Communication Systems (ICICS)*. IEEE, 2019, pp. 52–57.

[16] M. Graindl, "Zigbee 3.0 - ein neustart mit alten schwächen," Master's Thesis, FH St. Pölten, 2019.

[17] I. ZigBee Alliance, "Zigbee specification (2015)," ZigBee Alliance, Inc., Davis, CA, Standard, 08 2015.

[18] , "Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low rate wireless personal area networks (wpans)," *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, pp. 1–320, 2006.

[19] N. L. UK, "Maximizing security in zigbee networks," *Internet of Things Security Evaluation Series*, 2017. [Online]. Available: https://www.nxp.com/docs/en/supporting-information/MAXSECZBNETART.pdf

[20] D. I. Inc., "Zigbee networks," 2019, letzter Zugriff: 11.10.2021. [Online]. Available: https://www.digi.com/resources/documentation/Digidocs/90002002/Default.htm

[21] T. Zillner and S. Strobl, "Zigbee exploited: The good, the bad and the ugly," *Black Hat–2015 https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly.pdf(21.03. 2018)*, 2015.

[22] I. ZigBee Alliance, "Zigbee cluster library specification," ZigBee Alliance, Inc., San Ramon, CA, Standard, 01 2016.

[23] ——, "Zigbee cluster library specification," ZigBee Alliance, Inc., San Ramon, CA, Standard, 02 2018.

[24] ——, "Zigbee smart energy standard," ZigBee Alliance, Inc., San Ramon, CA, Standard, 12 2014.

[25] ——, "Zigbee specification (2008)," ZigBee Alliance, Inc., San Ramon, CA, Standard, 01 2008.

[26] X. Fan, F. Susan, W. Long, and S. Li, "Security analysis of zigbee," *Comput. Netw. Secur. Class, Massachusetts Inst. Technol., Cambridge, MA, USA, Rep*, 2017.

[27] S. Farahani, *ZigBee wireless networks and transceivers*. newnes, 2011.

[28] H. Lipmaa, P. Rogaway, and D. Wagner, "Ctr-mode encryption," in *First NIST Workshop on Modes of Operation*, vol. 39. Citeseer, 2000.

[29] T. Iwata and K. Kurosawa, "Omac: One-key cbc mac," in *International Workshop on Fast Software Encryption*. Springer, 2003, pp. 129–153.

[30] H. Li, Z. Jia, and X. Xue, "Application and analysis of zigbee security services specification," in *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, vol. 2, 2010, pp. 494–497.

[31] F. Kausar, S. Hussain, J. H. Park, and A. Masood, "Secure group communication with self-healing and rekeying in wireless sensor networks," in *International Conference on Mobile Ad-Hoc and Sensor Networks*. Springer, 2007, pp. 737–748.

[32] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O'Flynn, "Iot goes nuclear: Creating a zigbee chain reaction," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 195–212.

[33] M. S. Wara and Q. Yu, "New replay attacks on zigbee devices for internet-of-things (iot) applications," in *2020 IEEE International Conference on Embedded Software and Systems (ICESS)*, 2020, pp. 1–6.

[34] N. Vidgren, K. Haataja, J. L. Patino-Andres, J. J. Ramirez-Sanchis, and P. Toivanen, "Security threats in zigbee-enabled systems: vulnerability evaluation, practical experiments, countermeasures, and lessons learned," in *2013 46th Hawaii International Conference on System Sciences*. IEEE, 2013, pp. 5132–5138.

[35] X. Cao, D. M. Shila, Y. Cheng, Z. Yang, Y. Zhou, and J. Chen, "Ghost-in-zigbee: Energy depletion attack on zigbee-based wireless networks," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 816–829, 2016.

[36] S. Okada, D. Miyamoto, Y. Sekiya, and H. Nakamura, "New ldos attack in zigbee network and its possible countermeasures," in *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2021, pp. 246–251.

[37] G. H. Talakala and J. Bapat, "Detecting spoofing attacks in zigbee using device fingerprinting," in *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*, 2021.

[38] R. Cayre, F. Galtier, G. Auriol, V. Nicomette, M. Kaâniche, and G. Marconato, "Wazabee: attacking zigbee networks by diverting bluetooth low energy chips," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2021, pp. 376–387.

[39] R. M. Joshua Wright, Ryan Speers, "Killerbee api documentation," letzter Zugriff: 2022.

[40] M. Foundation, "Mariadb server," letzter Zugriff: 23.02.2022. [Online]. Available: https://mariadb.org/

[41] dresden elektronik ingenieurtechnik gmbh, "Phoscon homepage," letzter Zugriff: 27.02.2022. [Online]. Available: https://phoscon.de/de/

[42] Wireshark, "tshark - dump and analyze network traffic," 2021, letzter Zugriff: 02.08.2021. [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html

[43] N. L. UK, "Zigbee light link user guide," 2016. [Online]. Available: https://www.nxp.com/docs/en/user-guide/JN-UG-3091.pdf

[44] P. Biondi and the Scapy community, "Scapy - packet crafting for python2 and python3," letzter Zugriff: 27.10.2021. [Online]. Available: https://scapy.net/