Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Martin Landsmann

Entwurf und Implementierung einer Laufzeitüberwachung zur Absicherung des Routings mit RPL

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Martin Landsmann

Entwurf und Implementierung einer Laufzeitüberwachung zur Absicherung des Routings mit RPL

Masterarbeit eingereicht im Rahmen der Masterprüfung

im Studiengang Master Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Thomas C. Schmidt
Zweitgutachter: Prof. Dr. Martin Becke

Eingereicht am: 07.12.2017

Martin Landsmann

Thema der Arbeit
Entwurf und Implementierung einer Laufzeitüberwachung zur Absicherung des Routings mit RPL

Stichworte
IoT, RPL, RIOT, sicheres Routing, Sensornetzwerke

Kurzzusammenfassung
Die Masterarbeit analysiert Schwachstellen im IPv6 basierten Routingprotokoll für energiearme und verlustbehaftete Netzwerke RPL, die von Angreifern zur Beeinträchtigung der Routings oder zum Ausspähen von übertragenen Informationen ausgenutzt werden können. Dabei zeigt sich, dass ein Angreifer der alle Zugangsvoraussetzungen zur Kommunikation und Teilnahme in der Topologie erfüllt das Routingprotokoll ernsthaft gefährden kann. Mit den gewonnenen Erkenntnissen der Analyse wird verdeutlicht, dass die Absicherung der Schwachstellen von RPL möglich ist, jedoch für eine weitreichende Abdeckung systematisch und organisiert durchgeführt werden muss. Mithilfe der hier entwickelten und umgesetzten Laufzeitüberwachung für RPL, die zur Laufzeit Schwachstellen überwacht und das Verhalten des RPL-Knotens protokolliert, werden strukturierte Analysen von RPL und eine modulare Entwicklung von Gegenmaßnahmen ermöglicht.

Martin Landsmann

Title of the paper
Design and Implementation of Runtime Monitoring to Secure Routing with RPL

Keywords
IoT, RPL, RIOT, secure Routing, Sensornetworks

Abstract
The thesis analyzes vulnerabilities in the IPv6-based routing protocol for Low Power and Lossy Networks RPL, which can be exploited by attackers to compromise routing or eavesdrop transmitted information. It is shown that an attacker who meets all access requirements for communication and participation in the topology can seriously endanger the routing protocol. The outcome from the analysis make it clear that securing the vulnerabilities of RPL is possible, but needs to be systematically performed and well organized for extensive coverage. The runtime monitoring developed and implemented here, which observes vulnerabilities and logs the behavior of the RPL node, enables a structured analysis of RPL and modular countermeasure development.

Inhaltsverzeichnis

1 Einleitung

Das sogenannte Internet der Dinge (IoT) stellt einen Sammelbegriff dar, mit dem Geräte und Dinge zusammengefasst werden, die mit dem Internet in Verbindung stehen und darüber mit anderen Teilnehmern oder Servern kommunizieren. Dieser Begriff fällt häufig im Zusammenhang mit leistungsstarken Smartphones, Tablet-Computern und Kleingeräten die mit ähnlicher Hardware ausgestattet ist. Unter diesen Begriff fallen jedoch auch einfache Sensoren und Knoten denen nur eingeschränkte Rechenleistung und minimale Energieversorgung zur Verfügung steht. Die Kommunikation zwischen diesen Kleinstgeräten der Klasse 2 und darunter ist den Beschränkungen in der Leistungsfähigkeit der Knoten und ihrer eingeschränkten Energieversorgung unterworfen. Sollen diese kabellos miteinander kommunizieren, ein eigenständiges Netzwerk aufbauen und instand halten, müssen sie entsprechende Hardware und zugeschnittene Kommunikationsprotokolle einsetzten, die für ihren Handlungsspielraum konzipiert sind. Neben proprietären Lösungen für verfügbare Übertragungstechnologien und Protokolle etablieren sich in diesem Feld zunehmend offene und frei zugängliche Lösungen die auf offenen Standards beruhen. Die IEEE definierte mit dem 802.15.4 Standard eine offene und energieeffiziente Übertragungstechnologie zur Funkkommunikation in drahtlosen Sensornetzwerken. Sie dient den genannten IoT-Kleinstgeräten als gemeinsame Basistechnologie zur drahtlosen Kommunikation untereinander. Sie erlaubt ihnen ohne zusätzliche Software einfache Stern-Topologien für die Kommunikation untereinander aufzubauen. Um die Möglichkeit zu Schaffen die potenziell sehr große Anzahl an Geräten miteinander Kommunizieren zu lassen und jedem Gerät skalierbar eine eindeutige Adresse zu vergeben, ist eine IPv6 basierte Adressvergabe und Organisation der Teilnehmer unumgänglich. Da die Nutzung von IPv6 für die verfügbaren Ressourcen von IoT Kleinstgeräten unter Umständen nicht sinnvoll handhabbar ist, wurde von der *IPv6 Over LowPower and Lossy Networks* Arbeitsgruppe der Interner Engeneering Taskforce (IETF) das gleichnamige Verfahren 6LoWPAN[1] konzipiert. Sie definierten wie durch Komprimierung und Zusammenfassung der Informationen von IPv6 Paketen deren Größe auf eine für IoT-Kleinstgeräte handhabbare Größe reduziert werden kann. Dabei ist es für einen Knoten möglich zustandslos ein IPv6 Paket in ein 6LoWPAN Paket zu komprimieren und zurück zu einem IPv6 Paket umzuwandeln. Ausgestattet mit dieser zugeschnittenen Übertragungstechnologie als Basis zur Kommunikation, lassen sich die IoT-Kleinstgeräte untereinander und mit dem IPv6 basierten Internet verbinden. Dabei sind die mit teils heterogener Hardware ausgestatteten IoT-Kleinstgeräte den gleichen Bedingungen und Problemen ausgesetzt, wie sie aus typischen drahtlosen Netzwerken von Teilnehmern mit einem vielfachen an verfügbarer Rechenleistung und Energie, bekannt sind. Für die Organisation von IoT-Kleinstgeräten untereinander sind intakte Routen auf denen Informationen

[1] https://tools.ietf.org/wg/6lowpan/

zwischen den Knoten ausgetauscht werden zwingend erforderlich, speziell wenn die Teilnehmer nicht direkt mit ihrem Ziel Kommunizieren können und andere Teilnehmer als Router im Netzwerk zum Weiterleiten von Paketen nutzen müssen. Der Aufbau und die Instandhaltung einer robusten Routing-Infrastruktur ist dadurch eine Grundvoraussetzung für die gesicherte und zuverlässige Kommunikation zwischen den IoT-Kleinstgeräten. In dieser Arbeit wird das für Kleinstgeräte der Klasse 2 ausgelegte *Routing Protokoll for Lowpower and Lossy Networks* vorgestellt und hinsichtlich seiner Schwachstellen untersucht. Es ermöglicht dank seines Konzepts Topologien mithilfe Knoten in Baum Struktur aufzubauen und instand zu halten in denen die Knoten als Host und Router dienen. Es ermöglicht eine Topologie zielgerichtet auf einen Einsatzzweck zu parametrieren, und zu optimieren. Durch die verlustbehaftete drahtlose Kommunikation in einem IoT Netzwerk ist das Protokoll anfällig für Störungen. Die Störungen können aufgrund von Umwelteinflüssen, oder sich fehl verhaltenden Knoten, ausgelöst werden. Das Protokoll bietet dabei grundlegende Schutzfunktionen und Wartungsmechanismen zur Aufrechterhaltung einer funktionsfähigen Topologie. Angreifer können jedoch diese Mechanismen umgehen oder für Angriffe gegen einzelne Knoten oder die gesamte Topologie ausnutzen. Ausgehend von den analysierten und untersuchten Schwachstellen von RPL wird daraus ein Entwurf für eine Laufzeitüberwachung entwickelt. Mit ihrer Hilfe kann ein Knoten Zustandsänderungen aufgrund von empfangenen Nachrichten protokollieren, bewerten und dedizierte Schutzmaßnahmen gegen potenzielle Angriffe einleiten.

2 Grundlagen

2.1 RPL das Routingprotokoll für das Internet der Dinge

Die *Routing Over Low-power and Lossy Networks* (RoLL) Arbeitsgruppe der IETF[1] untersuchte die Kompatibilität und die Möglichkeit gängige Routing-Protokolle für das Routing auf IoT Geräten und in Sensornetzen einzusetzen [1]. Sie konnten kein Routing-Protokoll ermitteln, mit dem alle Kriterien für ein Routing auf Kleinstgeräten zufriedenstellend abgedeckt werden konnten. Mit den Erkenntnissen ihrer Evaluierung definierten sie Anforderungen für das Routing in 6LoWPAN [2] Netzwerken. Ausgehend von den definierten Anforderungen, spezifizierten sie das *Routing Protocol for Low power and Lossy Networks* (RPL) [3]. Die RPL Spezifikation definiert eine grundlegende Basis für das Routing in 6LoWPAN Netzwerken. Sie erlaubt gleichzeitig Spezialisierungen an individuelle Anwendungsszenarien in denen Topologien und Kommunikationsmuster zwischen den Teilnehmern auf ein bestimmtes Szenario abgestimmt sind [4], vorzunehmen. Ein Anwendungsszenario in dem RPL als Routing-Protokoll eingesetzt wird, kann abstrakt durch eine *Objective Function* (OF) beschrieben werden. Sie bestimmt, wie sich ein Knoten während des Topologieaufbaus verhält, und dadurch auch, wie sich die resultierende Form der Topologie herausbildet. Zukünftig sollen für abstrakt definierbare Szenarios individuelle *Request for Comments* (RFC) [5] entstehen, in denen auf das jeweilige Anwendungsszenario zugeschnittene OF definiert sind.
Unabhängig der spezifischen Anforderungen an Szenarien, in denen RPL als Routingprotokoll eingesetzt wird, wurden bereits zuvor RFCs mit allgemeinen Anforderungen für mögliche Einsatzszenarien in *Low-power and Lossy Networks* (LLN) definiert. RFC-5548 [6] definiert Anforderungen an ein LLN Routing-Protokoll für Anwendungen im Einsatz von Städten und Gemeinden. In RFC-5673 [7] werden Anforderungen zur Kontrolle von Automatisierungs- und Steuerungs-Anlagen im industriellen Bereich definiert. RFC-5826 [8] definiert Anforderungen für Heimautomatisierung und RFC-5867 [9] für Gebäudeautomatisierung.
Spezifisch für das RPL Routingprotokoll ist im RFC-6552 [10] die *OF Zero* (OF0) definiert. Die OF0 muss per Definition von jedem RPL Knoten unterstützt werden und erlaubt ihm, einen Pfad zum Root Knoten aufzuspannen, ohne Metriken in die Ermittlung des Pfades einbeziehen zu müssen. Dadurch wird jedem teilnehmenden Knoten der Topologie ermöglicht, eine Basistopologie aufzuspannen über die Kontrollnachrichten zum Root-Knoten weitergeleitet werden können. Eine weitere OF wurde in RFC-6719 [11] spezifiziert, die *Minimum Rank with Hysteresis OF* (MRHOF). Sie erlaubt es einem Knoten der RPL Topologie, einen Pfad zum Root-Knoten zu finden, der auf eine spezifische Metrik optimiert ist, bspw. auf eine minimierte Anzahl von Hops oder auf die erwartete Anzahl von erfolgreich übertragenen Paketen *Expected*

[1] https://datatracker.ietf.org/wg/roll

Transmission Count (ETX). RFC-6719 definiert dabei keine Metrik. Routing-Metriken, die mit MRHOF genutzt in 6LoWPAN Netzwerken eingesetzt werden, sind in RFC-6551 [12] definiert. Zusätzlich zu der Spezifikation von MRHOF, wird darin auch der Schutz und die Erkennung von falsch annoncierten Metriken diskutiert. Der Fokus liegt dabei auf der Beschreibung von möglichen Attacken gegen das Routing mithilfe von Manipulationen von Metriken, sowie der Möglichkeit die darauf basierenden Angriffe zu identifizieren.
Das RPL Routing-Protokoll ist für den Einsatz in *multihop*-LLNs ausgelegt. Das Protokolldesign von RPL sieht vor, dass das vorrangige Kommunikationsmuster einer RPL Topologie *Convergecast* ist. Dabei findet der Informationsfluss überwiegend von den Knoten der Topologie hin zu einem Root-Knoten statt. Neben dieser *Multipoint-to-Point* (MP2P) Kommunikation, erlaubt RPL auch eine *Point-to-Point* (P2P) Kommunikation zwischen zwei Knoten innerhalb der Topologie, sowie eine *Point-to-Multipoint* (P2MP) Kommunikation, in der bspw. der Root-Knoten Pakete gleichzeitig an mehrere Knoten in der Topologie verschicken kann.
Zum Aufbau von Routen, spannt RPL einen *Destination Oriented Directed Acyclic Graph* (DODAG) auf. In einem DODAG sind alle Pfade auf den Ursprungsknoten der Topologie, dem Root, ausgerichtet und spannen einen Baum mit ihm als Wurzel hin zu den entferntesten Knoten, den Blättern der Topologie auf. Ein DODAG ist eindeutig durch eine `DODAGID` identifiziert. Der Root eines DODAGs dient primär als Server zum Verarbeiten von eintreffenden Kontrollinformationen der Knoten in der Topologie, mit denen er den DODAG verwaltet und die Kommunikation zwischen den Knoten ermöglicht. Zusätzlich kann er die Kommunikation zwischen den Knoten im DODAG, sowie Zielen außerhalb des DODAGs in anderen IPv6 Netzwerken ermöglichen, wenn er Kontakt zu einem *6LoWPAN Borderrouter* (6LBR) hat oder selbst einer ist. Als grundlegende Basis im Aufbau eines DODAGs gilt die Regel, dass sich jeder teilnehmende Knoten strikt an eine hierarchische Ordnung beim Ermitteln seiner Position in der Topologie halten muss. Die Position eines Knotens innerhalb der Hierarchie wird durch seinen Rang definiert. Er wird von dem Knoten zusammen mit weiteren Informationen annonciert, und ermöglicht damit weiter entfernten Nachbarknoten, Teilnehmer des DODAGs zu werden. Ausgehend vom Root-Knoten, steigt der Rang jedes teilnehmenden Knotens im DODAG streng monoton mit seiner topologischen Distanz zum Root. Dabei wird durch die eingesetzte OF im DODAG festgelegt, anhand welcher Kriterien ein Knoten seine Eltern wählt und wie der Knoten seine topologische Distanz, den eigenen Rang, ermittelt.
Um die genannten Kommunikationswege MP2P, P2P und P2MP in einem DODAG aufzuspannen, müssen Routen sowohl in Richtung des Root-Knotens (*upward*) als auch in Richtung der Blattknoten (*downward*) etabliert werden. Hierfür definiert und nutzt RPL verschiedene Kontrollnachrichten, die beim Topologie-Aufbau sowie bei der Instandhaltung des DODAGs eingesetzt werden. Die Kontrollnachrichten sind:

- DIO (DODAG Information Object)
- DIS (DODAG Information Solicitation)
- DAO (Destination Advertisement Object)
- DAO-ACK (DAO Acknowledgment)

- CC (Consistency Check)

Sie werden in ICMPv6 Typ 155 Nachrichten[2] eingebettet und zwischen den Knoten der Topologie ausgetauscht.
Ein Knoten versendet DIO-Kontrollnachrichten, um seine Position im DODAG zu annoncieren und den Nachbarknoten in seiner Umgebung zu signalisieren, ob er als *default next-hop* Knoten zur Verfügung steht. Die DIO enthält dabei die benötigten Informationen, die einem Knoten ermöglichen dem DODAG als Teilnehmer beitreten zu können. Zusätzlich kann er damit die Präferenz zu einem bestimmten Elternknoten bestimmen, den er als bevorzugten Knoten zur Kommunikation *upward* in Richtung Root nutzt. Wird der Versender der DIO als neuer bevorzugter Elternknoten gewählt, berechnet der Empfänger-Knoten, unter Zuhilfenahme der Informationen aus der DIO und der OF, seinen eigenen Rang im DODAG. Durch das Versenden einer DIS-Kontrollnachricht kann ein Knoten eine DIO von seinen benachbarten RPL Knoten anfordern. DIO und DIS-Kontrollnachrichten dienen dazu, Kontrollnachrichten *upward* weiterzuleiten und Routen von den Blattkonten hin zum Root-Knoten aufzubauen.
Zum Aufbau von Routen in die entgegengesetzte *downward* Richtung werden DAO-Kontrollnachrichten eingesetzt. DAOs propagieren *upward* in Richtung des Root-Knotens und beinhalten Informationen zu den Zweigen eines Knotens in *downward* Richtung. Sie erlauben, Routen von einem Knoten aus hin zu den Blattknoten der Topologie aufzubauen. DAO-Kontrollnachrichten werden beim Empfang einer DIO versendet, um zu annoncieren welche Knoten der Versender einer DIO in *downward* Richtung erreichen kann. Ein Knoten verzögert dabei das Versenden der DAO-Kontrollnachrichten etwas. Dadurch kann der Knoten zunächst Informationen aus eintreffenden DAOs der *downward* Zweige konsolidieren, bevor er die Informationen in DAOs an seinen Elternknoten weiterleitet Der Empfang einer DAO, kann optional durch eine DAO-ACK-Kontrollnachricht bestätigt werden.
CC-Kontrollnachrichten werden zur Synchronisation von Zählern benachbarter Knoten verwendet. Die CC-Kontrollnachrichten bieten eine Grundlage für die Absicherung der Kommunikation zwischen den Knoten. Sie können Angriffe, die mithilfe von wieder eingespielter RPL Kontrollnachrichten durchgeführt werden, verhindern.

[2] http://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml#icmpv6-parameters-2

3 Gefährdungsszenarien und Angreifermodell

Angriffe auf ein System, haben die Absicht seine Kerneigenschaften zu kompromittieren, oder zu beschädigen. Die Schwachstellen eines Systems bieten die Angriffsfläche für den Angreifer, negativen Einfluss auf das System auszuüben. Ein Angreifer nutzt diese Schwachstellen aus und realisiert damit eine spezifische Gefährdung des Systems. Ein Gefährdungsszenario ist demnach die Realisierung eines Angriffs durch Ausnutzen einer oder mehrerer Schwachstellen eines Systems.
In den Gefährdungsszenarien werden Gefahren identifiziert, die eine Kompromittierung der Sicherheit im Routing zur Folge haben. Zur Evaluierung der daraus identifizierten Gefahren wird ein Angreifermodell definiert, mit dem die Umsetzbarkeit der Gefährdungsszenarien untersucht wird. Die diskutierten Gefährdungsszenarien in diesem Kapitel, basieren auf den Kategorien der Analysen aus RFC-7416 [13]. Jedes beschriebene Szenario ist einer der Kategorien Authentifizierung, Vertraulichkeit, Integrität und der Nicht-Abstreitbarkeit aus dem *ISO 7498-2 security reference model*[1] zugeordnet. Die in diesem Kapitel verwendeten Begriffe sind abgeleitet aus den Begriffsdefinitionen des RFC-2828 [14], sowie den untersuchten Bedrohungen gegen Routing Protokolle des RFC-4593 [15].

3.1 Terminologie

Autorisierter Knoten Beschreibt einen Knoten, der die Berechtigung hat auf eine geschützte Ressource zuzugreifen oder einen geschützten Dienst in Anspruch zu nehmen.

Kontrollebene Ist eine Differenzierung der Kommunikationsebene zwischen Knoten einer Topologie, in der Informationen beim Aufbau und der Instandhaltung der Topologie ausgetauscht werden. Router tauschen in der Kontrollebene Nachrichten mit Informationen zum Aufbau von Routen zu spezifischen Zieladressen aus. Knoten lernen in dieser Ebene die Router kennen und annoncieren ihre Zieladressen an sie.

Datenebene In dieser Differenzierung der Kommunikationsebene werden von den Knoten der Topologie Daten, die nicht der Kontrollebene angehören, an eine spezifische Zieladresse weitergeleitet. Router, die ein Paket der Datenebene empfangen, schlagen in ihrer Routingtabelle den nächsten Hop zum Weiterleiten des Pakets in Richtung der Zieladresse nach, den sie zuvor durch Nachrichtenaustausch in der Kontrollebene gelernt haben.

[1] http://www.iso.org/iso/catalogue_detail.htm?csnumber=14256

Täuschung Eine Aktion, die darin resultiert, dass ein autorisierter Knoten verfälschte Informationen als wahr annimmt. Eine Täuschung kann mittels Imitation einer autorisierten Herkunft der Informationen, oder durch das Verfälschen der Informationen in den übertragenen Daten der Nachricht erreicht werden.

- Imitation - Der Angreifer täuscht einen autorisierten Knoten, indem er die Informationen mit der Identität eines anderen autorisierten Knotens an ihn weiterreicht. Dadurch versucht er die Privilegien sowie die Glaubwürdigkeit über den Informationsinhalt von Nachrichten des imitierten Knotens, für seine versendeten Informationen zu beanspruchen. Damit versucht der Angreifer Zugriff auf geschützte Funktionen und Ressourcen zu erlangen, für die er keine Berechtigung hat. Dieser Angriff kann sowohl durch Wiedereinspielen von gelernten Informationen als auch durch Vortäuschen der Identität eines autorisierten Knotens durchgeführt werden.
- Verfälschung - Der Angreifer täuscht einen autorisierten Knoten mit gefälschten Daten. Hierfür kann er gefälschte Informationen generieren, oder versuchen gelernte Daten eines autorisierten Knotens zu manipulieren, bevor er diese an einen Knoten weiterleitet, der getäuscht werden soll.

Störung Ein Ereignis, dass den Betrieb und die Bereitstellung eines Dienstes hemmt oder den Dienst vollständig unterbricht.

- Beeinträchtigung - Der Angreifer stört den Betrieb des Dienstes, indem er eine Dienstkomponente blockiert.
- Korruption - Der Angreifer manipuliert den Ablauf im Betrieb des Dienstes, indem er Daten oder Funktionen verändert. Hierfür kann er die ausgetauschten Informationen in der Kontrollebene, oder die Funktionen des Dienstes für seine Zwecke manipulieren. Als Resultat wird die Logik des angegriffenen Dienstes verändert.
- Sabotage - der Angreifer unterbricht die Bereitstellung eines Dienstes. Hierfür blockiert er den Nachrichtenaustausch in der Kontroll- und Datenebene, oder erschöpft die Ressourcen von anderen Knoten, sodass sie den Dienst nicht mehr anbieten können.

Bespitzelung Ein Ereignis, bei dem der Angreifer direkten Zugriff auf vertrauliche Informationen erhält, für die er keine Zugriffsberechtigung hat.

- Abfangen - Der Angreifer belauscht den Nachrichtenaustausch zwischen autorisierten Knoten und wertet den konkreten Inhalt der übermittelten Nachrichten aus.
- Deduktion - Der Angreifer analysiert den Nachrichtenverkehr seiner Umgebung und ermittelt daraus Rückschlüsse auf die Charakteristik des Systems, bspw. den Aufbau der Topologie. Dabei wird der konkrete Inhalt von mitgehörten Nachrichten nicht ausgewertet.

Authentizität Die Herkunft einer Nachricht kann zweifelsfrei einer eindeutigen Quelle zugeordnet werden.

Integrität Die Unversehrtheit der übertragenen Informationen einer Nachricht kann eindeutig festgestellt werden.

Verfügbarkeit Ein Kriterium, mit dem ein Maß für die Erfüllung von festgelegten Anforderungen an einen Dienst definiert ist, das zu einem festgelegten Zeitpunkt oder innerhalb eines Zeitraums erfüllt werden muss.

Vertraulichkeit Die transportierten Informationen aus einer Nachricht können nur von dem Empfänger gelesen werden, für den sie bestimmt sind.

Gefährdung Die potenzielle Chance der Schädigung des Systems durch einen Angreifer. Der Angreifer nimmt hierfür durch das Ausnutzen von Schwachstellen Einfluss auf das System und seine Dienste.

3.2 Gefährdungsszenarien

Zu den gefährdeten Komponenten in einem LLN gehören die ausgetauschten und gespeicherten Informationen zum Routing sowie die verfügbaren Ressourcen der Knoten und die auf ihnen ausgeführten Prozesse. Die Informationen zum Routing werden in LLNs mittels drahtloser Kommunikation in der Kontrollebene ausgetauscht und teilweise auf den Knoten für die Weiterverarbeitung zwischengespeichert. Die Ressourcen auf den Knoten bestehen aus ihrer Rechenleistung, dem verfügbaren Speicher, ihrer bereitstehenden Energie, sowie der zur Verfügung stehenden Bandbreite für die Kommunikation mit ihren Nachbarn. In den Routing-Prozessen des Knotens sind Dienste zusammengefasst, mit denen Routen in der Topologie erzeugt und instand gehalten werden. Die Knoten eines LLN, erzeugen mittels drahtloser Kommunikation eine Kommunikationsinfrastruktur zum Aufbau einer Topologie in der Kontrollebene und nutzen sie zur Datenübertragung in der Datenebene. Durch die eingesetzte Funkübertragung von Nachrichten im LLN sind die ausgetauschten Informationen der Knoten anfällig, durch einen Angreifer abgefangen und manipuliert zu werden. Durch physischen Zugriff kann ein Angreifer Knoten manipulieren und erhält dadurch die Möglichkeit, Einfluss auf den Nachrichtenaustausch für das Routing eines LLNs zu nehmen. Zu den potenziellen Angriffen in unmittelbarer Umgebung der Topologie besteht die Möglichkeit Angriffe auf die RPL Knoten aus der Entfernung durch das Internet durchzuführen. Dies wird dadurch ermöglicht, weil RPL für den Einsatz auf ressourcenarmen Knoten entwickelt und dabei die Kommunikation und Interaktion mit Knoten aus anderen IPv6-basierten Netzwerken erlaubt. Für diese Arbeit wird die Annahme gemacht, dass ein Borderrouter, der eine RPL Topologie mit anderen IPv6-Netzwerken verbindet, hinreichend Schutzmechanismen einsetzt, um potenzielle Gefährdungsszenarien aus dem Internet gegen die Knoten zu unterbinden. Daher werden in

dieser Arbeit ausschließlich Gefährdungsszenarien aus dem unmittelbaren Einsatzbereich und Nähe einer RPL Topologie untersucht.

Gefährdung der Integrität und der Authentizität Der drahtlose Kommunikationskanal bietet einem Angreifer die Möglichkeit, in den Nachrichtenaustausch seiner Nachbarn einzugreifen. Er kann den Inhalt der von Nachrichten manipulieren, oder eigene verfälschte Nachrichten über das Kommunikationsmedium verbreiten. Ein Angreifer kann hierfür Nachrichten abfangen, die Informationen der Nachricht auswerten, den Inhalt manipulieren und abschließend die manipulierte Nachricht an seine Nachbarn weiterleiten. Hat der Angreifer einen autorisierten Knoten physikalisch kompromittiert, kann er auf dem gekaperten Knoten die Routinginformation direkt verändern, oder in den Programmablauf der Logik des Routings durch eigenen Code eingreifen. Durch das Kapern des Knotens erhält er seine Privilegien und Zugangsbefugnisse. Dadurch ist der Angreifer in der Lage, autorisierte Nachrichten zu versenden und verfälschte Routinginformationen an seine Nachbarknoten in der Kontrollebene zu verschicken. Ist es ihm nicht möglich einen Knoten zu täuschen, kann er versuchen einen anderen autorisierten Knoten zu imitieren, um seine Zugangsbefugnisse für Angriffe auszunutzen. Ist die Täuschung eines Knotens erfolgreich, erhält der Angreifer Zugriff auf die zugangsbeschränkten Ressourcen und Dienste des imitierten Knotens, die ihm für Angriffe auf das Routing zur Verfügung stehen.

Gefährdung der Verfügbarkeit Das Maß der Verfügbarkeit bezeichnet, ob ein Dienst festgelegte Anforderungen zu einem spezifischen Zeitpunkt oder innerhalb eines Zeitraums erfüllt. Ein Routing Protokoll dient zum Kreieren und Instandhalten von verlässlichen Routen innerhalb der Topologie. Auf diesen Routen, leiten die Knoten Informationen in der Kontrollebene und Daten in der Datenebene zu einer Zieladresse weiter. Benachbarte Router und Hosts der Topologie, tauschen hierfür Informationen in der Kontrollebene aus und beanspruchen dabei Ressourcen, um den Routing-Dienst für sich oder Nachbarknoten bereitzustellen. Werden die Ressourcen des Knotens erschöpft, oder die Prozesse für das Routing durch andere ausgeführte Prozesse blockiert, sinkt die Verfügbarkeit des Routings und die der Weiterleitung von Nachrichten in der Topologie. Störungen im Nachrichtenaustausch zwischen den Teilnehmern der Topologie gefährden die Verfügbarkeit von Routen und die Zuverlässigkeit des angebotenen Routing-Dienstes durch das Routing-Protokoll. Mithilfe von Korruption ausgetauschter Kontrollnachrichten, oder der Sabotage eines Dienstes auf dem Knoten, kann der Angreifer versuchen die Verfügbarkeit von RPL zu beeinträchtigen.

Gefährdung der Vertraulichkeit Für die Instandhaltung und den Aufbau einer Topologie werden potenziell vertrauliche und sensible Angaben für das Routing in der Kontrollebene zwischen den Knoten einer Topologie ausgetauscht. Sie sind durch ihren vertraulichen Inhalt, attraktiv und anfällig von einem Angreifer offengelegt und zu seinem Vorteil ausgenutzt zu werden. Die Kontrollnachrichten mit den vertraulichen Informationen kann er durch Mithören des Übertragungsmediums abfangen und auswerten. Sind die Informationen in den mitgehörten Nachrichten hinreichend verschlüsselt, sodass sie für den Angreifer nicht trivial auslesbar sind,

kann er ohne ihren Inhalt direkt zu verstehen, versuchen die Kommunikationsmuster seiner Umgebung zu analysieren. Hat er genügend Austausch von Kontrollnachrichten mitgehört, kann er durch Deduktion aus den Charakteristika der Nachrichtenflüsse, Schlüsse über Rollen in der Topologie und ihren Aufbau ziehen. Kann ein Angreifer erfolgreich einen autorisierten Knoten der Topologie kapern und physikalisch kompromittieren, hat er direkten Zugang zum Nachrichtenfluss der Kontroll- sowie Datenebenen der Topologie. Damit kann er die Informationen aus allen durch den kompromittierten Knoten geleiteten Nachrichten, für die er Zugriffsberechtigungen hat, lesen und auswerten.

3.3 Angreifermodell

Zum Schutz vor Angriffen definiert RPL optionale kryptografische Schutzmaßnahmen, die mithilfe von geheimen Schlüsseln eine gesicherte Kommunikation ermöglichen. Die Schutzmaßnahmen werden im Abschnitt 4.2 genau diskutiert. Durch die Möglichkeit optionale Schutzmaßnahmen in RPL einzusetzen, kann ein Angreifer in 2 Kategorien untergliedert werden, das Mitglied und der Außenstehende [16]. Grundsätzlich besitzen beide Kategorien des Angreifers die Fähigkeiten und Ressourcen, potenzielle Schwachstellen von RPL zu ihrem Vorteil für Angriffe auszunutzen.

Außenstehender Ein Angreifer dieser Kategorie ist ein nicht autorisierter Knoten, der versucht in die gesicherte Kommunikation von autorisierten RPL-Knoten einzugreifen, um Einfluss auf Knoten der Topologie auszuüben. Er ist dabei weder im Besitz der Zugangsvoraussetzungen, noch von kryptografischen Schlüsseln bei eingesetzter verschlüsselter Kommunikation, um als autorisierter RPL-Knoten im DODAG teilzunehmen. Der Angreifer hat Zugriff auf den drahtlosen Kommunikationskanal des LLNs, auf dem die RPL Knoten Nachrichten austauschen. Er ist mit leistungsfähiger Hardware ausgestattet und den Knoten der Topologie überlegen. Ihm stehen für Angriffe Komponenten zur Verfügung, die in einem nach derzeitigem Technikstand verfügbarem Laptop verbaut sind. Mit der Ausstattung hat er eine wesentlich höhere Rechenleistung sowie um Größenordnungen mehr verfügbaren Speicher gegenüber den angegriffenen Knoten der Topologie. Auf seinem Laptop hat er gängige Werkzeuge zur Analyse und zur Manipulation des Netzwerkverkehrs installiert, bspw. *Wireshark* und *Packeth*. Sein für die Kommunikation mit den RPL Knoten eingesetzter Funkchip hat eine ausgedehnte Reichweite für das Versenden und das Empfangen von Paketen. Diese übersteigt die Kommunikationsreichweite der angegriffenen Knoten. Mit seiner leistungsfähigen Hard- und Software kann der Angreifer den Nachrichtenaustausch in der Kontrollebene sowie der Datenebene seiner Umgebung mitlesen, analysieren und manipulierte Pakete an Knoten verschicken. Durch die hohe Reichweite seiner Sende- und Empfangsgeräte kann er physische Nähe zu anderen Knoten vortäuschen. Mit dem ausgedehnten Bereich, in dem der Angreifer den Nachrichtenaustausch der Topologie mithört, ist es ihm möglich, durch Deduktion einen Überblick über die Kommunikationsmuster und den Datenfluss der Topologie zu erhalten. Der Angreifer ist nicht auf ein einzelnes Gerät für seine Angriffe begrenzt. Er kann kollaborierende Geräte an

unterschiedlichen Stellen der Topologie einsetzen, die er über eine Nebenbandkommunikation seiner Wahl für Angriffe koordiniert.

Mitglied Ein Angreifer dieser Kategorie teilt zunächst die Charakteristik des außenstehenden Angreifers. Er hat jedoch, im Gegensatz zu diesem, alle Zugangsvoraussetzungen und Schlüssel zur authentifizierten Teilnahme und Kommunikation in der RPL Topologie. Dadurch ist er als ein autorisierter Teilnehmer im DODAG und kann Angriffe aus dem inneren der Topologie durchführen. Um die Schlüssel und Zugangsvoraussetzungen zu erhalten, kann ein Angreifer einen Knoten des DODAGs kompromittieren und dadurch Kontrolle über ihn und seine Zugangsberechtigungen erhalten. Hat der Angreifer physischen Zugriff auf einen unbeaufsichtigten Knoten, kann er den Speicher des Knotens auslesen und aus dem Speicherabbild seine Schlüssel und Zugangsberechtigungen extrahieren [17].

4 Sicherheitsevaluierung von RPL

In diesem Abschnitt wird die Sicherheit von RPL im Hinblick auf definierte Schutzziele untersucht. Als Grundlage der Untersuchung dient das zuvor in Abschnitt 3.3 eingeführte Angreifermodell. Die Untersuchung findet unter der Annahme statt, dass die RPL-Topologie vorab installierte Schlüssel auf den Knoten einsetzt, um die Kommunikation zu sichern. Die vorinstallierten Schlüssel werden gleichsam vom Root des DODAGs den Routern und Hosts eingesetzt. Ein Schlüsselaustauschverfahren, mit dem die Kommunikation zwischen den Knoten individuell kryptografisch abgesichert werden kann [18], wird hier bei Untersuchung außen vor gelassen. Durch den Einsatz eines Schlüsselaustauschverfahrens in RPL, kann der Wirkungsbereich durch einen vom Angreifer kompromittierten Knoten eingeschränkt werden. Dabei müssen individuelle Schlüssel zur verschlüsselten Kommunikation der Knoten im DODAG eingesetzt werden. Solange die Geheimhaltung der Schlüssel gewährleistet ist, können keine Angriffe zur Täuschung anderer Knoten mittels Imitation durchgeführt werden. Werden hingegen Gruppenschlüssel eingesetzt, die das Schlüsselaustauschverfahren periodisch durch neue Schlüssel ersetzt, ist der Angreifer in der Lage alle Angriffe auf die Mitglieder seiner zugehörigen Gruppe durchzuführen.
Für die Kategorie des außenstehenden Angreifers hat der Einsatz eines Schlüsselaustauschverfahrens gegenüber vorinstallierten geheimen Schlüsseln, keine wesentlichen Auswirkungen auf seine anwendbaren Angriffe. Da er keine Zugangsvoraussetzungen für die gesicherte Kommunikation im DODAG erfüllt, kann er versuchen, den Schlüsselaustausch zu stören oder durch Deduktion Schlüsse auf die Verteilungsstrategie der Schlüssel zu ziehen.
Für einen Angreifer, der bereits Mitglied und damit Teilnehmer der gesicherten Kommunikation im DODAG ist, schränkt das Schlüsselaustauschverfahren den Angriffsradius auf den Sicherungsbereich der gelernten Schlüssel und ihrer Befugnisse ein.

4.1 Schutzziele

Maßnahmen zum Schutz des Routings in RPL müssen die im Abschnitt 3.2 genannten Gefährdungsszenarien entschärfen, die ein Angreifer als Außenstehender oder Mitglied des DODAGs durch Angriffe realisiert. Im folgenden Teil werden für die aufgeführten Gefahren der genannten Kategorien Authentizität, Integrität, Verfügbarkeit und Vertraulichkeit konkrete Schutzziele festgelegt.

Authentizität Die Herkunft einer Nachricht muss einer bestimmten Quelle zugeordnet werden können, um ein Vertrauensverhältnis zwischen kommunizierenden Knoten eines LLNs aufzubauen. Daher muss RPL davor geschützt werden, dass ein Angreifer unberechtigt als

authentifizierter Knoten auftritt oder sich als ein anderer authentifizierter Knoten ausgeben kann, um verfälschte Informationen in dessen Namen zu verbreiten.

Integrität Ausgetauschte Nachrichten müssen gegen Manipulation abgesichert werden, sodass der Versuch einer Veränderung durch den Empfänger erkannt wird. Ein Knoten muss die Unversehrtheit der Informationen in einer Nachricht feststellen können. Mit dieser Fähigkeit können die Knoten Veränderungen der Informationen in empfangenen Nachrichten feststellen und dem Inhalt von unversehrten Nachrichten vertrauen. Dies unterbindet Angriffe, die Einfluss auf das Routing der Knoten durch Verfälschung der Informationen von mitgehörten veränderten und wieder eingespielten Nachrichten nehmen.

Verfügbarkeit Die Bereitstellung eines Dienstes und seiner Ressourcen erfordert, dass der angebotene Dienst für einen Einsatzzweck geeignet ist und zuverlässig funktioniert. Wird der Dienst gestört oder nachteilig für seinen Einsatzzweck beeinflusst ist die Verfügbarkeit des Dienstes bedroht. Zum Schutz vor dieser Gefahr muss RPL Ausfälle und Fehlverhalten von Knoten kompensieren können, indem hinreichend redundante Informationen in der Kontrollebene zur Aufrechterhaltung des Routings verteilt werden. Angriffe auf die Authentizität und die Integrität von RPL sind dabei häufig implizite Angriffe auf die Verfügbarkeit des Routing-Dienstes von RPL. Durch die Imitation oder Korruption eines autorisierten Knotens, in Kombination mit der Verfälschung von Informationen in ausgetauschten Kontrollnachrichten, kann der Angreifer das Routing beeinträchtigen. Schutzmaßnahmen zur Einhaltung der Authentizität und Integrität von Informationen tragen damit zum Schutz der Verfügbarkeit eines Dienstes bei. Eine einfachere Form der Sabotage des Routings ist das physikalische Stören des Übertagungsmediums. Dabei werden die Knoten im Einflussbereich des Angreifers an der Kommunikation gehindert [19]. Dies soll hier nur genannt werden, wird jedoch nicht näher untersucht, da Schutzmaßnahmen in der Vermittlungsschicht keine Handhabe gegen Angriffe in den darunter liegenden Schichten bieten.

Vertraulichkeit Nachrichten, die über einen unsicheren Kanal transportiert werden, sind anfällig, dass unautorisierte Dritte die Nachrichten mithören. Diese Teilnehmer haben damit Zugriff auf die transportierten Informationen und seinen vertraulichen Inhalt. Findet die Übertragung der ausgetauschten Nachrichten kabellos und verlustbehaftet statt, ist es schwierig festzustellen, ob die Nachricht von unautorisierten Dritten abgefangen und mitgehört werden konnte. Zur Wahrung der Vertraulichkeit muss der unberechtigte Zugriff auf den Inhalt der Nachricht durch Schutzmaßnahmen abgesichert werden. Üblicherweise wird hierfür der Inhalt der ausgetauschten Nachrichten verschlüsselt. Durch eine Analyse des reinen Kommunikationsflusses der Kontroll- und Datenebene kann ein Angreifer trotz eingesetzter Verschlüsselung Aufschluss über Pfade der Topologie erhalten. Die geschlossenen Erkenntnisse aus dieser Analyse stellen dabei keine direkte Gefährdung dar, können jedoch als Grundlage genutzt werden aussichtsreiche Ziele für Angriffe zu identifizieren.

4.2 Schutzmechanismen in RPL

RPL beinhaltet Maßnahmen zum Schutz von Kontrollnachrichten gegen Bespitzelung und Täuschung durch außenstehende Angreifer. Die Spezifikation sieht gesicherte Versionen aller Kontrollnachrichten sowie drei Modi zur Sicherung der Kommunikation innerhalb des DODAG vor.
Eine gesicherte RPL Kontrollnachricht wird durch das oberste Bit des `Code`-Feldes der zum Transport verwendeten ICMPv6 Nachricht identifiziert. Die so markierte ICMPv6 Nachricht beinhaltet zusätzliche Informationen, die zur Absicherung der beinhalteten RPL spezifischen Informationen dienen. Diese werden in einem *Security*-Abschnitt direkt hinter dem ICMPv6 Header und vor den Inhalt der RPL-Kontrollnachricht einfügt, vgl. Abb. 4.1.

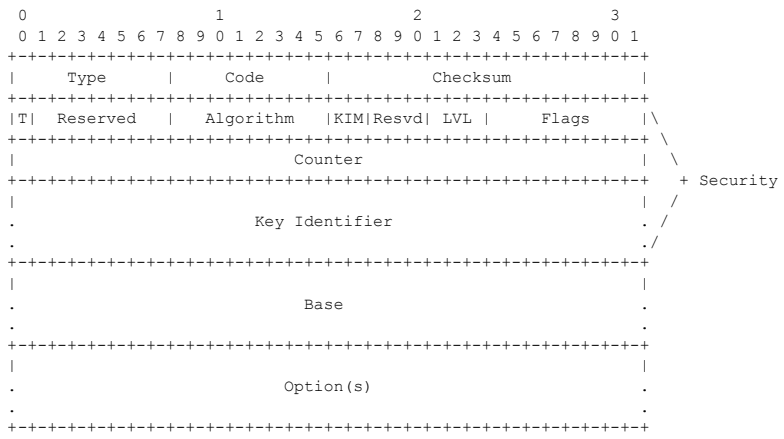

Abbildung 4.1: Aufbau einer gesicherten RPL-Kontrollnachricht

Der *Security*-Abschnitt beginnt mit dem T-Kontrollbit. Es zeigt an, ob die Kontrollnachricht mithilfe eines Zählers, oder eines Zeitstempels, gegen Verzögertes wieder einspielen abgesichert wird.
Gefolgt wird dieses Bit von weiteren 7 reservierten Bits für den zukünftigen Einsatz. Im darauf folgenden `Algorithm`-Feld wird festgelegt, welche Strategie zur Wahrung der Authentizität und Integrität der Kontrollnachrichten in der RPL Topologie eingesetzt wird. In der RPL Spezifikation ist einzig die Variante 0 definiert. Diese sieht vor, die Integrität von Kontrollnachrichten durch den Einsatz von AES-128 im CCM-Modus [20] mit 32-Bit *Message Authentication Codes* (MACs) zu sichern. Zur Gewährleistung der Authentizität von Kontrollnachrichten wird in der definierten Variante 0, eine digitale Signatur in Kombination aus RSA-Verschlüsselung mit SHA-256 eingesetzt.
Das darauf folgende *Key Identifier Mode* (`KIM`)-Feld definiert die Art des eingesetzten Schlüsselmanagements. Dabei kann festgelegt werden, ob Gruppenschlüssel, Schlüsselpaare oder digitale Signaturen von den Knoten für die Sicherung der Kontrollnachrichten eingesetzt werden.
Auf das KIM-Feld folgen 3 reservierte Bits für potenzielle zukünftige Einsatzzwecke. Darauf folgt das *Security Level*-Feld (`LVL`). Die drei Bits dieses Feldes erlauben, die Einstellungen zur

Art der eingesetzten Sicherung festzulegen. In Kombination mit dem Modus im KIM-Feld kann damit für jedes Paket individuell der Grad des Schutzes der Authentizität sowie der Vertraulichkeit einer Kontrollnachricht festgelegt werden. Es wird differenziert, ob 32-Bit oder 64-Bit MACs zur Authentifizierung, sowie ob 2048-Bit oder 3072-Bit lange RSA Signaturen eingesetzt werden.
Es folgt ein z.Zt. ungenutztes 1-Byte Flags-Feld für den potenziellen zukünftigen Einsatz. Das darauf folgende `Counter`-Feld beinhaltet abhängig von der Einstellung des T-Bits der *Security Level*-Feldes, einen Zählerstand, wenn es gesetzt ist, oder einen vom Versender der Kontrollnachricht eingetragenen Zeitstempel, wenn das Bit nicht gesetzt ist. Unabhängig der Charakteristik des eingetragenen Wertes, dient es dem Empfängerknoten veraltete oder wieder eingespielte Kontrollnachrichten zu erkennen.
Im darauf folgenden `Key Identifier`-Feld wird die Herkunft und der Index eines eingesetzten Schlüssels festgelegt. Die Herkunft des Schlüssels identifiziert einen Herausgeber von Schlüsseln, während der Index einen spezifischen Schlüssel des Herausgebers festlegt.
Im darauf folgenden Base-Abschnitt wird jeweils der Inhalt der RPL-Kontrollnachricht abgelegt, gefolgt von einer variablen Anzahl zusätzlicher Informationen im Option(s)-Abschnitt.
Die im *Security*-Abschnitt einer Kontrollnachricht festgelegten Sicherungsmaßnahmen werden auf das gesamte ICMPv6 Paket angewendet. So werden MACs und Signaturen über dem gesamten ungesicherten ICMPv6 Paket erzeugt. Veränderbare Felder des ICMPv6-Pakets, beispielsweise das `Checksum`-Feld, werden dabei für die Kalkulation temporär mit Nullen aufgefüllt. Hinterher werden die Werte in die entsprechenden Felder eingetragen. Bei eingesetzter Verschlüsselung wird der Kontrollnachrichteninhalt beginnend hinter dem *Security*-Abschnitt bis hin zum letzten Byte der des ICMPv6 Pakets verschlüsselt. Der Paketkopf sowie der *Security*-Abschnitt werden unverschlüsselt übertragen. Damit kann ein Empfänger die Header des Pakets lesen, und hat genügend Informationen, um eine gesicherte Kontrollnachricht zu verifizieren, und bei Bedarf entschlüsseln zu können.
RFC-6550 definiert, welche Voraussetzungen durch RPL-Knoten erfüllt sein müssen, um einer gesicherten RPL-Instanz beizutreten [21]. In RFC-6550 sind drei Sicherheitsmodi festgelegt, die bestimmen welche eingestellten Schutzmaßnahmen zur Sicherung von RPL Kontrollnachrichten Anwendung finden.

1. Der ungesicherte Modus legt fest, dass keine Schutzmaßnahme auf die Kontrollnachrichten angewendet wird. In diesem Modus wird in der Kontrollnachricht kein *Security*-Abschnitt hinzugefügt.

2. Der vorinstallierte Modus definiert, dass RPL Knoten, der als Host oder Router in der Topologie teilnehmen möchte, einen symmetrischen Schlüssel zum Beitreten einer RPL Instanz nutzen muss. Dieser dafür eingesetzte Schlüssel ist entweder vom Werk aus auf den Knoten vorinstalliert oder wird während einer Initialisierungsphase beim Aufbau einer Topologie auf die Knoten übertragen. Der Schlüssel dient gleichzeitig zur Authentisierung des Urhebers einer Kontrollnachricht, der Einhaltung ihrer Integrität sowie zur Gewährleistung der Vertraulichkeit beim Nachrichtenaustausch zwischen den Teilnehmern des DODAGs.

3. Der authentifizierende Modus definiert das Vorgehen, wie ein Knoten in einem DODAG gesichert die Rolle eines Elternknotens übernehmen kann. Ein RPL Knoten kann mit einem vorinstallierten Schlüssel gesicherte RPL Kontrollnachrichten auswerten und seinerseits versenden, sodass er dem DODAG als Host beitreten kann. Der Schlüssel erlaubt dem Knoten, sich bei einer Schlüsselautorität anzumelden und einen anderen kryptografischen Schlüssel anzufordern. Die Schlüsselautorität soll dabei die Authentifizierung und Autorisierung des RPL Knotens durchführen. Der neu bezogene Schlüssel von der Schlüsselautorität, erlaubt es dem Knoten sich als Router gegenüber anderen Knoten auszuweisen. Damit kann er die Rolle eines Elternknotens im DODAG annehmen. Die RPL Spezifikation definiert, dass der authentifizierte Modus durch asymmetrische kryptografische Verfahren geschützt werden muss. Sie spezifiziert dabei nicht wie die Schlüsselautorität alle geforderten Schutzziele beim Schlüsselaustausch und der Legitimation von Knoten für die Rolle eines Elternknotens einhalten soll.

4.3 Kryptografische Schutzmaßnahmen von RPL

RPL definiert den Einsatz von kryptografischen Verfahren zum Authentifizieren von ausgetauschten Kontrollnachrichten, ihrer Verschlüsselung und dem Schutz vor Wiedereinspielen veralteter Informationen. Diese sind durch die einstellbaren kryptografischen Schutzmaßnahmen des *Security*-Abschnitts einer RPL-Kontrollnachricht individuell einstellbar. Sie finden jedoch keine Anwendung zur Sicherung von RPL spezifischer Informationen in Paketen der Datenebene. Dadurch liegt der untersuchte Fokus in diesem Abschnitt auf den verfügbaren Schutzmaßnahmen des Nachrichtenaustauschs in der Kontrollebene.

Schutz vor Angreifern außerhalb des DODAGs Ein außenstehender Angreifer hat ohne die nötigen Schlüssel, keine Möglichkeit eine gesicherte Kontrollnachricht zu entschlüsseln oder einen gültigen MAC zu erzeugen, mit dem eine Manipulation von abgefangenen Kontrollnachrichten verschleiert werden kann. Für einen erfolgreichen Angriff auf die Integrität ausgetauschter Informationen der Kontrollebene des DODAGs, muss er versuchen, den zur Sicherung eingesetzten MAC der seinerseits durch eine AES-128 Verschlüsselung geschützt ist, zu brechen.
Die Sicherheit von CCM wurde in [22] evaluiert. Dort wurde gezeigt, dass ein Angreifer ohne den Zugang zu dem geheimen Schlüssel einer verschlüsselten Nachricht weder erfolgreich die Informationen ermitteln, noch eine eigene valide Verschlüsselung kreieren kann. Zum Entschlüsseln von AES-128 ohne Zugriff auf den eingesetzten Schlüssel ist kein algorithmischer Ansatz bekannt, der weniger Zeit in Anspruch nimmt als die erschöpfende Suche. Der außenstehende Angreifer muss zum Entschlüsseln einer AES-128 gesicherten Kontrollnachricht den richtigen Schlüssel aus einem Schlüsselraum von 2^{128} Raten, was nicht in einer handhabbaren Zeitspanne möglich ist [23, 24]. Durch den eingesetzten und verschlüsselten MAC sichert RPL die Kontrollnachrichten gegen Verfälschung des Inhalts und damit gleichzeitig gegen Imitation der Identität eines anderen Knotens.
Alternativ zu CBC-MACs ist es in RPL möglich RSA Signaturen einzusetzen. Die Authentifikati-

on und Verifikation der Nachricht ist analog zum Einsatz von MACs, mit dem Unterschied, dass asymmetrische Verschlüsselung eingesetzt wird. Die Sicherheit von RSA Signaturen wurde in [25] evaluiert. Dort wurde gezeigt, dass das RSA Problem [26] generell schwierig zu lösen ist, und für Schlüssellängen von mindestens 1024 Bits kein Verfahren bekannt ist, mit dem die Suche relevant beschleunigt [27] werden kann. Ein außenstehender Angreifer kann unter diesen Umständen weder eine gültige RSA Signatur erzeugen, noch den geheimen Schlüssel aus mitgehörten verschlüsselten Nachrichten ermitteln. Dies verhindert, dass er erfolgreich andere Knoten durch Imitation oder Verfälschung einer Nachricht täuscht und die so gesicherten Kontrollnachrichten abfangen kann.
Durch den Einsatz einer digitalen Signatur zum Absichern der Kontrollnachricht kann der Urheber einer Kontrollnachricht mithilfe von Zertifikaten eindeutig zugeordnet werden. Zum Verifizieren der eingetragenen Signatur muss jedoch eine Infrastruktur für die Verteilung von Zertifikaten existieren, auf der Knoten die Zertifikate zur Singnaturprüfung anfordern können. Derzeit ist hierfür keine standardisierte Lösung für RPL definiert. In RPL sind die Schlüssellängen für RSA auf 2048 oder 3072 Bits festgelegt, wodurch ein Knoten der die Sicherungsmaßnahmen einsetzt, über ausreichend Speicher für deren Berechnung und für die Singnaturprüfung verfügen muss. Die asymmetrische Verschlüsselung einer Datenmenge ist gegenüber einer symmetrischen Verschlüsselung der gleichen Daten immer speicherintensiver [28]. Dies bringt eine erhöhte zu übertragene Menge an Daten beim Nachrichtenaustausch zwischen den Knoten mit und hebt damit ihren Energieverbrauch an [29, 30].
Durch den Einsatz von asymmetrischen RSA Signaturen oder symmetrische CBC-MACs in RPL werden Angriffe durch einen außenstehenden Angreifer, bei denen Kontrollnachrichten verfälscht werden oder sich der Angreifer, als ein autorisierter Knoten ausgeben kann, verhindert.

Schutz vor Angreifern innerhalb des DODAGs Ein Angreifer, der ein als autorisierter Knoten im DODAGs teilnimmt, kann kryptografisch gesicherte Kontrollnachrichten im DODAG empfangen und die beinhalteten Informationen lesen. Er ist gleichermaßen in der Lage, valide und durch die Schutzmaßnahmen gesicherte Kontrollnachrichten an andere Knoten zu versenden. Dadurch wird es ihm ermöglicht, falsche Informationen in seinen Kontrollnachrichten einzubringen und zu verteilen. Zudem kann er die durch ihn geleiteten RPL Kontrollnachrichten mit gefälschten Informationen anreichern oder den Inhalt vollständig austauschen, falls ihm die dafür erforderlichen Schlüssel zur Verfügung stehen. Der Einsatz von paarweiser Verschlüsselung zwischen kommunizierenden Knoten kann dabei den Wirkungsbereich des Angreifers einschränken, sodass Manipulationen von weitergeleiteten oder wieder eingespielten Kontrollnachrichten sehr aufwendig oder sogar, wie für den außenstehenden Angreifer, unmöglich werden.

4.4 Schutz vor dem Wiedereinspielen von Kontrollnachrichten in RPL

Beim Einsatz von gesicherten Kontrollnachrichten enthält RPL Schutzmaßnahmen die vor dem Wiedereinspielen der Nachrichten schützen. Dies wird durch den Einsatz von Sequenzzählern in den Kontrollnachrichten geschützt. Die böswillige Verzögerung beim Ausliefern von Kontrollnachrichten wird mithilfe von Zeitstempeln unterbunden.
Zum Schutz vor wieder einspielen von Kontrollnachrichten werden *nonces* generiert und als Zähler in das Counter-Feld im *Security*-Abschnitt einer gesicherten Kontrollnachricht eingetragen. Ein *nonce* wird initial mithilfe von CCM erzeugt, und erhält dadurch die Eigenschaft sich nicht zu wiederholen wenn beliebig weitere *nonces* mit CCM generiert werden. Ein Knoten verwendet für die gesicherte Kommunikation mit einem anderen RPL Knoten in der Kontrollebene zwei Sequenzzähler, die mit jeweils einem *nonce* initialisiert wurden. Einer der Zähler wird für den Schutz ausgehender Kontrollnachrichten eingesetzt, der andere für die Verifikation von eingehenden Kontrollnachrichten. Beim Versenden einer gesicherten Kontrollnachricht erhöht der Knoten immer den eingetragenen *nonce* des Sequenzzählers für ausgehende Kontrollnachrichten. Dies findet immer individuell für die Kommunikation zu einem Knoten statt, sodass jeder Nachrichtenaustausch mit unterschiedlichen Initialwerten und Zählerständen beginnt. Beim Empfang einer gesicherten Kontrollnachricht wertet der Knoten den Zählerstand des eingetragenen *nonce* aus. Er überprüft dabei, ob der Zählerstand höher als sein gespeicherter Sequenzzähler für eingehende Nachrichten vom Versender ist und erhöht ihn entsprechend. Ist der *nonce* kleiner oder gleich dem geführten Sequenzzähler für eingehende Kontrollnachrichten, ist der Inhalt der Kontrollnachricht veraltet und potenziell sogar ein Angriffsversuch durch Wiedereinspielen. Der Knoten verwirft dann die empfangene Kontrollnachricht. Solange die Sequenzzähler der kommunizierenden Knoten synchronisiert sind, kann zuverlässig ein Angriffsversuch durch Wiedereinspielen von Kontrollnachrichten aufgedeckt und verhindert werden.
Zur Synchronisation der Zähler werden CC-Kontrollnachrichten eingesetzt. Der Austausch von CC-Kontrollnachrichten wird initiiert, wenn eine explizite Anfrage für eine CC-Kontrollnachricht auf dem Knoten eintrifft, oder der eingetragene *nonce* einer gesicherten Kontrollnachricht 0 ist, obwohl der Knoten einen Zählerstand ungleich 0 für den Versender führt. Zum Schutz der Re-Synchronisierung von Zählerständen wird in die Anfrage für eine CC-Kontrollnachricht ein zusätzlicher *nonce* eingetragen. Die erwiderte CC-Kontrollnachricht mit den Zählerständen muss diesen *nonce* enthalten, um den Sequenzzähler des angefragten Knotens zu aktualisieren. Im Gegensatz zu der bewusst eingeleiteten Anfrage für eine CC-Kontrollnachricht kann beim Empfang einer gesicherten Kontrollnachricht mit dem Zählerstand von 0 kein Schutz vor Wiedereinspielen der CC-Kontrollnachricht gewährleistet werden. Dies resultiert aus dem dann trivial zu erratenden nonce. Die Sequenzzähler können keinen Schutz vor verzögertem Weiterleiten veralteter gesicherten Kontrollnachrichten bieten. Dabei wird eine empfangene gesicherte Kontrollnachricht durch den außenstehenden Angreifer abgefangen und zu einem späteren Zeitpunkt an Knoten versendet. Zum Schutz vor diesen Angriffen kann RPL einen Zeitstempel in das Feld für den CCM-Zählerstand in gesicherten Kontrollnachrichten eintragen.

Übersteigt die Zustelldauer einer gesicherten Kontrollnachricht ein festgelegtes toleriertes Limit, wird die Kontrollnachricht verworfen. Die Manipulation des eingetragenen Zeitstempels durch einen außenstehenden Angreifer wird durch die eingetragene Prüfsumme über den Inhalt der Kontrollnachricht verhindert. Alternativ können Zeitstempel zum Schutz vor Wiedereinspielen von gesicherten Kontrollnachrichten verwendet werden. Hierfür wird jedoch der Einsatz eines Zeitsynchronisationsverfahrens durch die RPL Knoten vorausgesetzt.

4.5 Themenverwandte Arbeiten

Die im DODAG eingesetzten Modi zum Sichern der Kontrollnachrichten bieten eine Basis zum Schutz gegen externe Angriffe. Risiken durch Angriffe innerhalb eines DODAG sowie Gegenmaßnahmen zum Schutz sind nicht durch die Spezifikation abgedeckt. Die Erkennung und Protokollierung von Angriffen innerhalb eines DODAGs wird in den nachfolgend vorgestellten Dokumenten näher untersucht. Dort werden potenzielle Angriffe und deren Auswirkungen auf die RPL-Topologie näher betrachtet.
In [13] analysieren die Autoren offene und nicht durch die bisherigen RFCs behandelte Aspekte im Zusammenhang mit der Sicherheit in RPL. Sie beschreiben Gefährdungsszenarien sowie Angriffe gegen das Routing und ordnen diese thematisch den Kategorien Authentifizierung, Vertraulichkeit, Integrität und nicht Abstreitbarkeit aus dem *ISO 7498-2 security reference model*[1] zu.

In der Kategorie Authentifizierung legen sie fest, dass ein RPL-Knoten seine Routing-Partner authentifizieren muss, bevor er sie in Routen als next-hop nutzt. Kann dieses Schutzziel nicht eingehalten werden, wird es einem Angreifer möglich authentifizierte Knoten des DODAGs zu imitieren oder fiktive Knoten im DODAG zu annoncieren.

In der Kategorie Vertraulichkeit müssen die RPL Knoten gewährleisten, dass Informationen zum Routing nur mit legitimierten Nachbarknoten ausgetauscht und nur von solchen verarbeitet werden können. Wird dieses Schutzziel nicht eingehalten, kann ein Angreifer Informationen zum Aufbau und der Instandhaltung der Topologie mithören und durch ihre Auswertung potenziell wichtige Knotenpunkte identifizieren.

Die Kategorie Integrität verlangt, dass ausgetauschte Routing Informationen zwischen RPL-Knoten gegen eine Verfälschung des Inhalts geschützt sein müssen. Wird dieses Schutzziel gebrochen, kann ein Angreifer durch die Manipulation des Inhalts von Kontrollnachrichten direkten Einfluss auf den Aufbau und die Instandhaltung der Topologie nehmen und das Routing sabotieren.

Das Schutzziel der Kategorie der Nicht-Abstreitbarkeit erfordert, dass versendete Informationen zum Routing direkt sowie retrospektiv einer eindeutigen Quelle zugeordnet werden können. Das nicht einhalten dieses Schutzziels, ermöglicht dem Angreifer manipulierte Routing-Informationen im Namen anderer Knoten des DODAGs in die Topologie einzuspielen. Die Einhaltung dieses Schutzziels erfordert jedoch vom LLN-Knoten, empfangene Routing Informationen zwischenzuspeichern und einer eindeutigen Quelle zuzuordnen. Da diese Anforderung eine sehr hohe Speichernutzung zur Folge hat, wird das Schutzziel von ihnen nicht weiter als

[1] http://www.iso.org/iso/catalogue_detail.htm?csnumber=14256

ein zwingendes Kriterium zum Absichern des Routings in LLNs gefordert.
Zusätzlich zu den betrachteten Kategorien des *ISO 7498-2 security reference model*, stellen die Autoren die Anforderung, die Verfügbarkeit von Routing-Informationen zum Sichern von RPL als Schutzziel einzuhalten. Bei nicht Einhaltung dieses Schutzziels kann ein Angreifer, bspw. durch Zurückhalten von Kontrollnachrichten, den reibungslosen Betrieb des Routings in einer RPL Topologie stören. Dadurch beeinflusst er das Konvergieren von Routen im DODAG, und beeinträchtigt damit seine Funktionstüchtigkeit sowie das angestrebte Ziel der Topologie.
Die Autoren diskutieren größtenteils Attacken durch einen außenstehenden Angreifer. Sie geben Empfehlungen, wie die Angriffe auf den OSI-Schichten 2 und 3 durch bekannte Schutzmaßnahmen, wie das *Transport Layer Security Protocol* (TLS) [31], abgeschwächt oder verhindert werden können. Sie schlagen als generelle Vorkehrung zum Schutz, eine gesicherte Kommunikation auf der Schicht 2 einzusetzen. In dieser würden dann entweder Gruppenschlüssel oder individuelle Schlüssel zwischen kommunizierenden Knoten eingesetzt werden. Angriffe aus dem Inneren des DODAG, klassifizieren die Autoren als byzantinische Ausfälle [32]. Zum Schutz vor solchen Angriffen empfehlen die Autoren, gesicherte RPL-Kontrollnachrichten auf der Vermittlungsschicht einzusetzen und eintreffende Pakete hinsichtlich der Häufigkeit sowie ihren Inhalt zu untersuchen. Mit der kritischen Beobachtung und Analyse von Kontrollnachrichten und Daten-Paketen können laut den Autoren Angriffe, aus denen Inkonsistenzen der Topologie entstehen, potenziell erkannt werden.
In [33] befassen sich die Autoren mit Angriffen, die unter Zuhilfenahme von Manipulationen des Rangs von RPL-Knoten durchgeführt werden. Sie untersuchen die nachteiligen Auswirkungen solcher Attacken auf die RPL Topologie, im Hinblick auf zuvor festgelegte *Quality of Service* (QoS) Eigenschaften, bspw. die Verzögerung der Zustellung von Paketen während eines Angriffs. Für die Anfälligkeit von RPL, den Rang für Angriffe ausnutzen zu können, identifizieren die Autoren Einschränkungen der Spezifikation. Demnach muss ein RPL Kind-Knoten Informationen aus den Kontrollnachrichten eines *default next-hop* Knotens verarbeiten und berücksichtigen, kann die gelernten Informationen jedoch nicht überprüfen. Als Konsequenz daraus propagieren verfälschte, gelernte Abhängigkeiten der Rangfolge eines Zweigs ungehindert *downward* im DODAG.
In [34] stellen die Autoren ein Überwachungssystem für RPL vor, dass durch dedizierte Knoten die Kommunikation im DODAG beobachten und protokollieren kann. Die überwachenden Knoten werden physikalisch in der Topologie verteilt und können als RPL-Knoten im DODAG teilnehmen. Sie verfügen einerseits über eine Sende- und Empfangseinheit zur Kommunikation mit RPL-Knoten im DODAG, und andererseits über zusätzliche Hardware für die Kommunikation mit den anderen überwachenden Knoten. Ausgestattet mit den zusätzlichen Sende- und Empfangseinheiten, formen die überwachenden Knoten eine zum überwachtem DODAG unabhängige Topologie. Die separat aufgebaute Topologie wird von den überwachenden Knoten genutzt, um Aufzeichnungen über den DODAG an einen Server weiterzuleiten. Der Server konsolidiert und analysiert die empfangenen Informationen über den DODAG. Zum Protokollieren und Verarbeiten von mitgehörten Paketen im DODAG verfügen die überwachenden Knoten, gegenüber den reinen RPL-Knoten im DODAG, über leistungsstärkere Sende- und Empfangseinheiten mit denen sie einen erweiterten Sende- und Empfangsradius im DODAG abdecken. Durch die Nutzung einer unabhängigen Topologie entkoppeln die Autoren den

Nachrichtenaustausch der überwachenden Knoten von dem des DODAGs. Damit wollen die Autoren vermeiden die eingeschränkte Bandbreite im LLN und Ressourcen den überwachten RPL-Knoten mit zusätzlichen Kommunikationsaufkommen zu belasten. Von den Autoren durchgeführte Simulationen zeigen, dass der Server mithilfe der analysierten Überwachungsinformationen, Inkonsistenzen sowie Anomalien im DODAG erkennen kann.

Weitere Literatur behandelt zusätzlich Strategien sowie Präventiv- und Gegen-Maßnahmen, zum Schutz vor Angreifern im Inneren eines DODAGs. Die folgenden Dokumente widmen sich einer Zusammenfassung von bekannten Angriffen, und potenziellen Gegenmaßnahmen die Angriffe zu unterbinden oder abzuschwächen.
In [35] diskutieren die Autoren Angriffe gegen RPL und ordnen diese in 3 Hauptkategorien *Ressources*, *Topology* und *Traffic* ein.
In der Kategorie *Ressources*, fassen die Autoren Angriffe zusammen, mit denen der Energieverbrauch von RPL-Knoten angehoben wird. Dabei wird durch den Angreifer versucht, einen Knoten mit rechenintensiven Operationen zu belasten und seinen Speicher häufig zu belegen. Die Motivation hinter den dabei aufgeführten Angriffen ist, die Leistungsfähigkeit der gesamten Topologie zu schwächen und dadurch den Betrieb im DODAG zu stören. Die Autoren führen hierfür drei konkrete Angriffe auf.
(i) Bei der *HELLO flood* Attacke, zwingt ein Angreifer seine Nachbarknoten kontinuierlich DIO Nachrichten an ihn zurück zu versenden. Für diese Attacke versendet der Angreifer periodisch DIS-Kontrollnachrichten an die Multicast-Adresse `ff02::01a`, die von allen Nachbarknoten seines Sendebereichs empfangen und verarbeitet wird. In RFC-6550 ist spezifiziert, dass der Empfang einer DIS-Kontrollnachricht auf dieser Multicast-Adresse, als lokale Inkonsistenz vom Empfänger gewertet wird. Zum Auflösen einer lokalen Inkonsistenz setzt ein Knoten zunächst seinen *trickle-timer* [36] für den DODAG zurück. Der *trickle-timer* bestimmt auf einem Knoten, die Länge der Pausen zwischen dem periodischen Versenden von DIO-Kontrollnachrichten. Durch das Zurücksetzen vom *trickle-timer*, beginnt der Knoten sofort und dann in ansteigenden Zeitabständen, DIOs an die Multicast-Adresse zu versenden. Alle Nachbarn des Angreifers, die seine DIS-Kontrollnachricht auf der Multicast-Adresse empfangen haben, beteiligen sich gleichermaßen an der Auflösung der Inkonsistenz, indem sie in aufsteigenden Zeitabständen periodisch DIOs versenden. Im Sendebereich des Angreifers erhöht sich dadurch direkt das Aufkommen von versendeten DIO-Kontrollnachrichten auf dem Übertragungsmedium. Bei Empfang der DIO-Kontrollnachrichten werden sie von den jeweiligen Knoten verarbeitet. Für diese zusätzliche Verarbeitung von eingehenden DIOs und das Versenden der Kontrollnachrichten, beansprucht ein Knoten seine Ressourcen und erhöht damit seinen Energieverbrauch. Wird der Angriff periodisch fortgesetzt, bewirkt das erhöhte Aufkommen der versendeten Kontrollnachrichten eine stetige Belegung des Übertragungsmediums. Dies kann einerseits Paketkollisionen provozieren, und andererseits dazu führen, dass ein Knoten während der Verarbeitung der vom Angreifer ausgelösten Kontrollnachrichten, andere Kontrollnachrichten verpasst. Wodurch echte Inkonsistenzen hervorgerufen werden können.
(ii) Im *Storing Mode of Operation* (Storing MOP) kann ein Angreifer seine *default next-hop* Knoten dazu bringen, ihre lokalen Routing-Tabellen mit falschen Einträgen zu befüllen. Hierfür versendet der Angreifer eine DAO an jeden Elternknoten, in der er Präfixe einträgt, die er

vermeintlich in seinem sub-DODAG erreichen kann. Füllt der betroffene Elternknoten seine Routing-Tabelle mit diesen Präfixen vollständig auf, kann er unter Umständen keine weiteren Präfixe aus den sub-DODAGs anderer Kind-Knoten verwalten. Der Angreifer versucht damit zu erreichen, dass Präfixe der sub-DODAGs von Knoten unterhalb seiner angegriffenen Elternknoten, nicht mehr *upward* propagieren und bekannt werden. Ist der Angriff erfolgreich, kann er dazu führen, dass zu den betroffenen Präfixen keine Routen *downward* aufgebaut werden und der DODAG degeneriert.
(iii) Ein Angreifer kann den Austausch von Kontrollnachrichten im DODAG erzwingen, indem er künstlich lokale Inkonsistenzen herbeiführt. Hierfür missbraucht er die Instandhaltungsmaßnahmen von RPL, mit denen Inkonsistenzen im DODAG lokal für benachbarte Knoten und global für alle Knoten im DODAG aufgelöst werden. Lokale Inkonsistenzen werden aufgelöst, indem die Knoten in der Umgebung einer lokalen Inkonsistenz, DIO-Kontrollnachrichten austauschen und mit den ausgetauschten Informationen die inkonsistenten Eltern-Kind Beziehungen und damit die monotone Rangfolge im DODAG reparieren. Zum trivialen Erzeugen einer Inkonsistenz, kann ein Angreifer versuchen das Übertragungsmedium zu stören, sodass keine Kommunikation zwischen Knoten stattfinden kann, oder vermehrt Paketkollisionen entstehen. Vom Angriff betroffene Knoten werden ihre Eltern-Kind Beziehungen nach hinreichend langer Dauer des Angriffs auflösen, und letztendlich versuchen, neue Elternknoten zur Kommunikation mit dem Root zu finden. Gelingt es dem Angreifer die Eltern-Kind Beziehungen kontinuierlich zu stören, wird der reibungslose Informationsaustausch in der Kontrollebene beeinträchtigt. Aufgrund dieses Angriffs werden lokale Reparaturen eingeleitet und Kontrollnachrichten zwischen den betroffenen Knoten ausgetauscht. Zusätzlich wirkt sich der Angriff direkt auf die Paketweiterleitung in der Datenebene aus, da die Knoten durch fortwährendes Löschen und neu lernen von Routen keine zuverlässige Weiterleitung der Datenpakete im DODAG sicherstellen können.
Bei einer fortgeschritteneren Variante des Angriffs verfälscht der Angreifer periodisch den eingetragenen Rang in seinen DIO-Kontrollnachrichten. Trägt er einen attraktiven Rang ein, kann er Nachbarn seiner Umgebung als Kinder gewinnen, die ihre Routen auf ihn lenken. Verschlechtert er seinen Rang hingegen, suchen sich die Kinder andere, attraktivere, Elternknoten und richten ihre Routen auf sie aus. Im einfachsten Szenario für den Angriff oszilliert der Angreifer periodisch zwischen einem niedrigen Attraktivem und einem hohen, weniger attraktiven Rang. Dadurch richten seine Nachbarknoten ihre Routen fortwährend neu aus, was einerseits den Austausch von Kontrollnachrichten anhebt, und andererseits die Stabilisierung gelernter Routen verhindert.
Um den Einfluss eines Angriffs dieser Kategorie im DODAG auf alle Knoten auszuweiten, kann der Angreifer unerlaubt die DODAG-Versionsnummer in seinen versendeten Kontrollnachrichten erhöhen. Eine erhöhte und damit neuere DODAG-Versionsnummer hat zur Folge, dass Knoten die sich in einer veralteten Version befinden, neue Elternknoten aus der annoncierten DODAG Version suchen. Letztendlich werden ausgehend vom Angreifer, alle Eltern-Kind Beziehungen zwischen den Knoten im DODAGs aufgelöst. Alle Knoten im DODAG, mit der Ausnahme des Root, werden dadurch gezwungen Kontrollnachrichten auszutauschen mit denen sie den DODAG neu aufbauen. Dies wirkt sich erheblich auf die Performanz in der Kommunikation zwischen den Knoten im DODAG aus [37, 33]. Sind die Knoten im DODAG

batteriebetrieben, verringert der zusätzlich aufkommende Austausch von Kontrollnachrichten in den aufgeführten Angriffen, ihre Lebensdauer und verkürzt die Gesamtdauer, in der eine zuverlässige Kommunikation im DODAG gewährleistet werden kann.
In der Kategorie *Topology* diskutieren die Autoren Angriffe, mit denen die Ausrichtung der Pfade eines DODAGs manipuliert werden kann. Hierfür versucht der Angreifer, gezielt den DODAG zu deformieren, sodass die Pfade auf denen Nachrichten ausgetauscht werden, gezielt in bestimmte Abschnitte des DODAGs gelenkt oder der Nachrichtenaustausch auf den betroffenen Zweigen gestört wird. Im *Storing MOP* kann der Angreifer durch manipulierte DAOs, fiktive Präfixe annoncieren und damit seinen Elternknoten Pfade zu nicht existierenden sub-DODAG Knoten vortäuschen. Die Information über die annoncierten Präfixe der DAO, propagiert *upward* zum Root des DODAG. Alle Elternknoten im betroffenen Zweig tragen die annoncierten Präfixe in ihre Routing-Tabellen ein, und können potenziell keine sub-DODAGs weiterer Kinder eintragen und bedienen. Knoten, deren *downward* Präfixe nicht im betroffenen Zweig bedient werden können, haben nur noch die Möglichkeit, falls vorhanden, ihre Präfixe in anderen Zweigen *upward* zu annoncieren. Migrieren die Knoten ihre Pfade zu den alternativen Elternknoten, werden ihre *downward* Routen aus dem betroffenen *upward* Zweig des Angreifers verdrängt, was zu einer Degenerierung des ursprünglichen DODAGs führt. Schließt sich fälschlicherweise ein Kind mit seinem sub-DODAG einem angegriffenen Elternknoten an, der den sub-DODAG nicht mit Nachrichten in *downward* Richtung bedienen kann, werden die Präfixe nicht *upward* annonciert und bleiben auf dem *downward* Pfad zu dem betroffenen Elternknoten unbekannt. Alternativ kann der Angreifer versuchen unter Zuhilfenahme der Manipulation seines annoncierten Rangs, Einfluss auf die Ausrichtung der Pfade im DODAG zu nehmen. Hierfür annonciert der Angreifer einen falschen, niedrigen Rang an seine Nachbarn. Das steigert seine Attraktivität gegenüber den Nachbarknoten ihn als Elternknoten zu nutzen und ihre *upward* Routen auf ihn auszurichten. Wählt er einen hinreichend attraktiven Rang, kann er letztendlich die Routen seiner Nachbarn und ihrer sub-DODAGs auf sich lenken. Ausgehend von der resultierenden zentralen Position, wird ein Großteil der Pakete zu ihm weitergeleitet. Durch die günstige Position im DODAG kann er Pakete mitlesen oder sie bspw. selektiv verwerfen und ein *Sinkhole* in der Topologie erzeugen. Kollaborieren mehrere Angreifer, können sie mit den beschriebenen Mitteln, gezielt Bereiche oder den gesamten DODAG von der Kommunikation mit dem Root ausschließen.
In einer passiven Variante des Angriffs nutzt der Angreifer die Eigenschaft der verlustbehafteten Kommunikation in LLNs aus, und versucht die Paketverlustrate in der Kommunikation zwischen Root und den Knoten seines sub-DODAGs zu maximieren. Dabei verhält er sich ausschließlich regelkonform und wählt lediglich die von ihm aus am schlechtesten erreichbaren Elternknoten zur Kommunikation mit dem Root, bei der möglichst wenige Pakete ihr Ziel unbeschadet erreichen.
In der Kategorie *Traffic* beschreiben die Autoren Angriffe, in denen Pakete durch den Angreifer mitgehört, abgefangen und manipuliert werden. Die Analyse des mitgehörten Datenverkehrs ermöglicht dem Angreifer, Aufschluss über den DODAG und Informationen über die übermittelten Daten seiner Umgebung zu erlangen. Dadurch kann er unter Umständen den Zustand der Topologie in seiner Umgebung ermitteln, ohne Teil des DODAGs zu sein. Ist der Angreifer Teil des DODAGs kann er durch gezielte Veränderung von RPL-Kontrollnachrichten Identitäten

benachbarter Knoten imitieren, oder neue fiktive Knoten in den DODAG annoncieren.
Die Autoren sammeln und ordnen bekannte Präventivmaßnahmen aus der Literatur gegen die Angriffe der festgelegten Kategorien zu. Die dort genannten konkreten Maßnahmen, sowie weitere vorgeschlagene Maßnahmen zum Schutz aus der Literatur werden, im weiteren Verlauf dieses Kapitels näher erörtert.
In [4] und [38] führen die Autoren eine Sicherheitsanalyse durch, in der sie die Protokollschichten von IoT-Netzwerken betrachten. Dabei untersuchen sie Protokolle, die in offenen Standards definiert sind. Als gängiges Routing-Protokoll für das IoT untersuchen sie RPL hinsichtlich seiner Sicherheit. In ihrer Analyse, stellen sie den unzureichenden Schutz in RPL gegen Angreifer die bereits Mitglied im DODAG sind fest. Sie skizzieren die Ansätze von [39] und [40] als potenzielle Ansätze zur Absicherung von RPL, die hier im weiteren Verlauf erörtert werden. Als offenen Punkt hinsichtlich der Sicherheit in RPL, identifizieren die Autoren den nicht näher spezifizierten Schlüsselaustausch.

In [39] skizzieren die Autoren eine Attacke, in der ein angreifender Knoten den Platz des Root-Knotens im DODAG annehmen kann. Im beschriebenen Szenario ist es einem Angreifer möglich, einen beliebigen Rang vorzugeben und damit die attraktive Position des DODAG-Roots zu imitieren. Annonciert der Angreifer den attraktiven Rang des Root-Knotens als seine topologische Position, löst er eine Rekonstruktion im DODAG aus. Seine Nachbarknoten, die nicht den tatsächlichen Root als Elternknoten nutzen, wählen daraufhin den Angreifer als ihren neuen bevorzugten Elternknoten. Als Konsequenz der Rekonstruktion wird einen Großteil der Routen im DODAG auf den Angreifer gelenkt. Die daraus resultierende zentrale Position im DODAG ermöglicht dem Angreifer, weitere Angriffe gegen die durch ihn geleiteten Pakete auszuführen oder sie für weitere Angriffe zu nutzen. Die Autoren schlagen als vorbeugende Maßnahme ein Verfahren vor, mit dem die DODAG-Versionsnummern und die annoncierten Ränge von Knoten, gegen unerkannte Manipulationen abgesichert werden. Dabei setzten sie Einweg-Hash-Funktionen, MAC-Authentisierung und digitale Signaturen zum Schutz von DIO-Kontrollnachrichten ein. In dem vorgeschlagenen Verfahren wird zuerst eine maximale Anzahl an DODAG-Versionsnummern festgelegt. Daraufhin wird eine Hashkette mit der festgelegten Anzahl der Versionen berechnet. Als initialen Eingangswert für die Hashkette wählt der Root einen geheimen *nonce*. Eine Hashkette wird durch die Iteration von Hashvorgängen konstruiert, in der in jedem Iterationsschritt der Letzte resultierenden Hashwert als neuer Eingangswert der Hashfunktion genutzt wird. Jedes Element der resultierenden Hashkette dient als eindeutiges Identifikationselement einer DODAG-Versionsnummer. Daraufhin werden zwei weitere Hashketten mit jeweils unterschiedlichen *nonce*s berechnet, deren Elemente jeweils einen spezifischen Rang identifizieren. Während der bootstrapping Phase, wird das zuletzt berechnete Endelement der Hashkette für die DODAG-Versionsnummern, sowie das zuletzt berechnete Endelement einer der Hashketten für den Rang signiert in einer Nachricht an alle Knoten verteilt. Zusätzlich erhält die signierte Nachricht die Iterationstiefe der verteilten Endelemente, sowie einen MAC der über dem Endelement der zweiten Hashkette für den Rang bestimmt wird. Für den MAC dient das vorletzte Element der Hashkette für die DODAG-Versionsnummern als Schlüssel. Das als Schlüssel eingesetzte Element der Hashkette ist bis dahin nur dem Root des DODAGs bekannt.

Nach Abschluss des Bootstrappings verifizieren alle Knoten die verteilten Endelemente der Hashketten mithilfe der Signatur und nutzen diesen Vertrauensanker zur Verifikation der strengen monotonen Folge von annoncierten Rängen und neueren DODAG-Versionsnummern in Kontrollnachrichten. Um die Verifikation eines annoncierten Rangs zu ermöglichen, wird in dem vorgeschlagenen Verfahren, der Rang, als Tupel vom Wert der Iterationstiefe mit dem zugehörigen Element der Hashkette für den Rang, in DIO-Kontrollnachrichten eingetragen. Beginnend mit dem Root, versendet dieser DIOs mit einem Tupel aus dem ersten Element zusammen mit seiner Iterationstiefe des Elements, der zurzeit im DODAG verwendeten Hashkette für den Rang. Dieses Element ist bis dahin nur dem Root bekannt, und ermöglicht seinen Nachbarknoten zu verifizieren dass der annoncierte Rang in der DIO streng monoton ansteigend berechnet und gewählt wurde. Zur Verifikation ermittelt der Knoten zunächst die Differenz aus der Iterationstiefe der signierten Hashkette für den Rang und der annoncierten Iterationstiefe der empfangenen DIO-Kontrollnachricht. Mit der ermittelten Differenz als Iterationszahl berechnet der Knoten das Endelement einer Hashkette, in die er das annoncierte Element des Tupels für den Rang aus der DIO als Eingangswert einsetzt. Stimmt der resultierende Hashwert mit dem signierten Endelement der Hashkette für den Rang überein, ist der annoncierte Rang verifiziert. Hat sich der Knoten dem Root als sein Kind angeschlossen, bestimmt er sein Tupel für den Rang, indem er das Nachfolgeelement der Hashkette für den Rang zu der des Root und die entsprechend erhöhte Iterationstiefe einträgt. Dieses Tupel nutzt er in seinen DIO-Kontrollnachrichten als seinen Rang. Dadurch ist es den Nachbarn eines Knotens möglich, seinen annoncierten Rang zu verifizieren. Analog dazu generieren sie eigenen Tupel, mit dem wiederum ihr Rang verifiziert werden kann. Damit werden Veränderungen des Rangs eines Knotens, auf seine vom Standard erlaubten Anpassungen eingeschränkt. Dabei gilt die Einschränkung, dass ein Knoten jeden Rang größer oder gleich des niedrigsten gelernten Elements der aktuellen Hashkette für den Rang annehmen kann.
Die DODAG-Versionsnummer wird bei dem vorgestellten Verfahren, unter Zuhilfenahme des beim Bootstrapping signiert verteilten Endelements der DODAG-Versionsnummer Hashkette abgesichert. Dabei nutzen die Autoren bei dem vorgeschlagenen Verfahren aus, dass nur der Root Knoten die Vorgängerelemente kennt und ein *preimage attack* auf einen Hashwert hinreichend schwierig erfolgreich durchgeführt werden kann. Konkret verteilt der Root zum Erhöhen der DODAG Versionsnummer ein Tupel, aus dem bis dahin nur ihm bekannten Vorgängerelementes des aktuell verteilten Elements für die DODAG-Versionsnummer im DODAG zusammen mit dem Abstand an Iterationen zum signierten Endelement aus dem Bootstrapping. Ein Knoten bestimmt mit dem Abstand an Iterationen, das Endelement einer Hashkette in die er das empfangene Vorgängerelement als Eingangswert einsetzt. Gleicht das resultierende Element dem im Boostrapping verteilten Endelement für die DODAG Versionsnummer legitimiert dies ein Erhöhen der aktuellen DODAG Version. Zusammen mit diesem Tupel verteilt der Root einen MAC über dem Endelement der Rank-Hashkette für die darauf nachfolgende DODAG-Version, sowie dem Endelement der Rang-Hashkette für den aktuellen DODAG Versionswechsel. Wechselt der Knoten in die neu annoncierte DODAG Version, verwirft er zuerst sein aktuell genutztes Endelement der Hashkette für den Rang. Das Endelement zum Verifizieren des Rangs der neuen DODAG Version, ist über den MAC aus dem vorherigen DODAG Versionswechsel gesichert. Zur Verifikation des Endelements bestimmt der Knoten

einen MAC, für den er den Hash aus dem legitimierten Endelement der neuen DODAG Version als Schlüssel nutzt. Stimmt der ermittelte MAC mit dem MAC der vorherigen DODAG Version überein, ist das empfangene Endelement des Rangs für die neue DODAG Version verifiziert. Im ersten Versionswechsel ist der MAC durch die Signatur aus dem Bootstrapping geschützt und kann durch sie verifiziert werden. Folgende Versionswechsel machen sich die Eigenschaft zunutze, dass der Vertrauensanker aus dem Bootstrapping über die Endelemente der Hashketten für die DODAG Version bestehen bleibt.
Durch geschicktes Zurückhalten von Kontrollnachrichten eines Versionswechsels, kann ein Angreifer das vorgeschlagene Verfahren aushebeln. Die dabei ausgenutzte Schwäche des Verfahrens ist der fehlende Bezug der verteilten Rang-Hashketten zum ursprünglichen Vertrauensanker aus der signierten Nachricht vom Bootstrapping. Ein Angreifer kann diese Schwäche ausnutzen, um beliebige legitime Elemente einer Hashkette für den Rang erzeugen. Empfängt der Angreifer eine legitime DIO-Kontrollnachricht vom Root, die einen Versionswechsel verursacht, versucht er den Versionswechsel vor seinen Nachbarknoten zurückzuhalten. Gelingt ihm dies, kann er mit dem gelernten Hashketten-Element der DODAG-Versionsnummer vom Root, einen legitimen MAC über einem ihm frei wählbaren Rank-Hashketten Endelement zu erzeugen. Damit kann er eine eigene Hashkette für den Rang berechnen und verteilen und in der DIO-Kontrollnachricht platzieren. Kann er darauf nachfolgend erneut eine DODAG-Versionserhöhung zurückhalten, ist er in der Lage, sein Element der frei gewählten Hashkette für den Rang zu verteilen. Dieses Element wird daraufhin durch einen Knoten erfolgreich verifiziert und als legitim betrachtet. Das ermöglicht dem Angreifer, einen beliebigen Rang gegenüber seinen Nachbarknoten vorzugeben. In [41, 42] wird diese Schwäche des Verfahrens untersucht und repariert. Für die vorgeschlagene Reparatur werden zunächst alle Rang-Hashketten Elemente für alle DODAG-Versionen vorab berechnet. Dann wird das Rang-Hashketten Endelement der vorletzten DODAG-Version symmetrisch verschlüsselt. Als Schlüssel dient das Endelement der Hashkette für den Rang aus der letzten DODAG-Version. Der resultierende Geheimtext dient dann als Schlüssel, um das Endelement der Hashkette für den Rang der DODAG Vorgänger-Version zu verschlüsseln. Die Endelemente werden nach diesem Verfahren so oft iterativ verschlüsselt, bis das Element der zweiten DODAG-Version mit dem Endelement der Hahskette für den Rang aus der dritten DODAG-Version zu einem Geheimtext verschlüsselt wird. Als Ergänzung zur signierten Nachricht aus dem Bootstrapping des ursprünglichen Verfahrens wird dieser Geheimtext zur Nachricht hinzugefügt. Zusätzlich wird bei jedem Versionswechsel, der Geheimtext zur nachfolgenden DODAG-Version verteilt. Mit diesem Geheimtext als Schlüssel kann im aktuellen Wechsel der DODAG-Version der verteilte Geheimtext des letzten Versionswechsels entschlüsselt werden. Der Klartext dient als Endelement der Haschkette für den Rang für die aktuell eingeleitete DODAG-Version. Das vorgeschlagene erweiterte Verfahren ermöglicht einem Knoten, die annoncierten Informationen zur DODAG-Verion sowie dem annoncierten Rang aus einer empfangenen DIO zu validieren.
In [43] haben die Autoren Attacken untersucht, in denen ein Angreifer unter Zuhilfenahme von Manipulationen des Rangs *Sinkhole*s im DODAG erzeugen kann. Die Autoren stellten bei ihren Simulationen und Messungen fest, dass ein Angriff, der ein *Sinkhole* erzeugt, durch die Beobachtung der Häufigkeit des Austauschs von DIO Nachrichten erkennbar ist. Auch die Autoren von [40] diskutieren die Auswirkungen von *Sinkhole*-Attacken. Sie untersuchen nachteilige

Effekte auf die Performanz bei der Zustellung von Nachrichten Ende-zu-Ende, während der DODAG einem Angriff ausgesetzt wird, der in einem *Sinkhole* in der Topologie resultiert. Die Autoren schlagen ein Verfahren vor, in dem ein Knoten den Verlust der Konnektivität zum Root des DODAGs über einen Elternknoten feststellen kann. Dabei nutzt das vorgestellte Verfahren zunächst ein an [39] angelehntes Verfahren zur Verifikation eines annoncierten Rangs von DIOs, mit dem die erfolgreiche Täuschung von Nachbarknoten bezüglich des Rangs erschwert wird. Zusätzlich führen die Autoren eine Ausfallsicherung von RPL Elternknoten ein, die ihnen erlaubt festzustellen, wenn der Root keine Verbindung zu ihnen hat. Die Ausfallsicherung der Konnektivität zum Root über einen Elternknoten wird durch ein Ende-zu-Ende Bestätigungsverfahren durchgeführt. Hat ein Knoten über eine festgesetzte tolerierte Zeitspanne den Root-Knoten nicht kontaktiert, speichert dieser die ID des vermissten Knotens in einer Liste ab. Die Liste wird periodisch an alle Knoten im DODAG verteilt. Findet sich ein Knoten in der Liste wieder, obwohl er die erwarteten Nachrichten zum Root-Knoten versendet hat, verwirft er seinen aktuell genutzten Elternknoten. Zusätzlich sperrt er ihn lokal als möglichen zukünftigen Elternknoten. Simulationen durch die Autoren zeigen, dass die Kombination der Verifikation des Rangs sowie der Ausfallsicherung, zuverlässig die Auswirkungen von *Sinkhole*s aber auch natürlich auftretender Inkonsistenzen abmildert.
In [44] wird ein Verfahren vorgestellt, dass den kompletten Pfad zwischen dem Root des DODAG und einem beliebigen Knoten im DODAG, auf eine monoton ansteigende Abfolge des Rangs hin validieren kann. Ausgehend von einem Knoten im DODAG, kann hierfür eine rekursive Pfadvalidierung ausgelöst werden. Sie bezieht alle Knoten im *upward* Pfad bis zum Root in den Validierungsprozess ein. Für das vorgeschlagene Verfahren, wird eine einzige Nachricht eingesetzt die *upward* bis zum Root propagiert und dann von ihm signiert zurück zum Knoten *downward* versendet wird. Die eingesetzte Nachricht beinhaltet die aktuelle DODAG-Versionsnummer des Knotens, der die Validierung auslöst und eine Liste in der spezifische Informationen jedes Knotens zu seinem individuellem Rang im *upward* Pfad kodiert. Hierfür nutzt das vorgeschlagene Verfahren die Eigenschaft von Bloom-Filtern [45], in denen probabilistisch die Präsenz von kodierten Informationen abgefragt werden kann, sowie die Möglichkeit die Filter zerstörungsfrei zu vereinen.
Ein Knoten, der sich am Validierungsprozess beteiligt, wählt für die Pfadvalidierung einen *nonce*, den er in einen Bloom-Filter einträgt. Den *nonce* trägt er als erstes Element einer Liste an Bloom Filtern in die Validierungsnachricht ein und versendet die Nachricht an seinen bevorzugten Elternknoen. Beim Empfang einer Validierungsnachricht überprüft ein Knoten zuerst die monotone Ordnung zwischen dem Rang des Versenders und seinem eigenen. Ist die streng monotone Abfolge gegeben, überprüft er die Aktualität der eingetragenen DODAG-Versionsnummer in der Validierungsnachricht. Sie darf in *upward* Richtung nur von einem Knoten mit größerem Rang angenommen werden und Versender sowie Empfänger der Kontrollnachricht müssen sich in der gleichen DODAG-Version befinden. Ist eine dieser Bedingungen verletzt, wird die Nachricht durch den Empfänger verworfen. Sind beide Bedingungen erfüllt, stellt der Knoten seinen Bloom-Filter als erstes Element der Liste von Bloom-Filten in der Validierungsnachricht voran und versendet seinerseits die Nachricht weiter an seinen bevorzugten Elternknoten. Treffen mehrere Validierungsnachrichten in *upward* Richtung bei einem Knoten aus seinen *downward* Zweigen ein, vereinigt dieser alle eingetragenen Bloom-Filter ausgerichtet nach

dem Rang der Versender, bevor er seinen Bloom-Filter als erstes Element der Liste voranstellt. Durch die Konstruktion der Liste von Bloom-Filten aller Knoten im Validierungspfad, sind in jedem Element der Liste alle *nonces* jeweils einer Rangebene zusammengefasst. Die Indexposition eines Elements entspricht dabei dem Rang einer Ebene. Das Verfahren wird sukzessive fortgesetzt bis die bei jedem Hop erweiterte Validierungsnachricht zum Root des DODAG propagiert. Hat der Root-Knoten die eingetroffene Nachricht verarbeitet, schickt er sie signiert zurück in Richtung der Blätter des DODAG. Die Signatur erlaubt es einem Knoten festzustellen, dass die Validierungsnachricht vom Root verarbeitet und anschließend zurückgesendet wurde. Ist dies der Fall, kann ein Knoten überprüfen, ob sein auf dem Hinweg eingetragener *Nonce* im Element der Liste von Bloom-Filtern an der Indexposition seines Rangs eingetragen ist. Wurde die monotone Ordnung der Ränge im DODAG eingehalten, kann der Knoten aufgrund der Konstruktion der Liste in *upward* Richtung seinen *nonce* wiederfinden. Ein Knoten, der unerlaubt einen beliebig attraktiveren Rang als seinen tatsächlichen annonciert, wird aufgrund der Konstruktion der Liste und der anschließenden Validierung durch seine Nachbarknoten direkt erkannt. Versucht ein Knoten unerlaubt die DODAG-Versionsnummer zu erhöhen, löst er aufgrund der Neubestimmung des Rangs eines Knotens in der neuen DODAG-Version, automatisch eine Pfadvalidierung aus. Erst eine signierte Nachricht vom Root legitimiert den Wechsel in die neue DODAG-Version. Tests durch die Autoren haben gezeigt, dass ein Knoten der einen falschen Rang annonciert letztendlich im DODAG isoliert wird und keinen Nachbarn erfolgreich täuschen kann. Durch das Verfahren werden *Sinkhole*s, die mithilfe von Manipulationen des Rangs durchgeführt werden verhindert. Zusätzlich schützt das Verfahren vor unerlaubtem Erhöhen der DODAG-Versionsnummer.
In [46] untersuchen die Autoren Attacken gegen RPL in denen ein Angreifer ein *Sinkhole* erzeugt. Als Gegenmaßnahme stellen sie ein Angriffserkennungssystem auf Basis eines Herzschlag-Protokolls vor. Diesen Ansatz greifen sie in [47] auf, um selektives Weiterleiten und verwerfen von Paketen durch einen Angreifer aufzudecken. In dem dort vorgestellten Verfahren rekonstruiert ein 6LBR des DODAGs eine vollständige Karte der Topologie. Die Karte erlaubt ihm, Aufschluss über Inkonsistenzen im DODAG aufzudecken und ermöglicht potenziell angreifende Knoten, die ihren Rang manipulieren, zu identifizieren. Das vorgestellte Verfahren bedarf grundsätzlich einer gesicherten Kommunikation zwischen den Knoten, die mittels IPSEC [48] oder DTLS [31] gesichert wird. Die erforderlichen Informationen zur Rekonstruktion der Topologie werden von den Knoten im DODAG somit gesichert an den 6LBR versendet. Aus den Informationen kann der 6LBR die Verteilung von Knoten sowie ihren expliziten Eltern Kind Beziehungen im DODAG ermitteln. Hat der 6LBR hinreichend Informationen über die Knoten im DODAG gesammelt, kann er Inkonsistenzen im DODAG bestimmen. Die Latenz, bis Schlüsse aus den eintreffenden Informationen gezogen werden können, steigt dabei abhängig mit der Anzahl von Knoten und der Tiefe des DODAGs. Zum Absichern der übermittelten Pakete mit den Informationen gegen Identifikation durch einen Angreifer schlagen die Autoren vor den 6LBR so zu konfigurieren, dass er für jeden Knoten der Topologie eine individuelle IPv6-Adresse zum Sammeln der Informationen bereitstellt. Damit wird es einem Angreifer, der erfolgreich einen Knoten kompromittiert hat, erschwert durch ihn geleitete Informations-Pakete trivial zu erkennen und ggf. verwerfen zu können. Jede erkannte Inkonsistenz wird vom 6LBR im rekonstruierten DODAG protokolliert. Für

jeden Knoten, der als Urheber einer Inkonsistenz identifiziert wurde, wird dabei ein Zähler erhöht. Überschreitet der Zählerstand einen vorab eingestellten Schwellwert, wertet der 6LBR die Inkonsistenz als Angriff eines Knotens gegen die Topologie. Der eingestellte Schwellwert muss spezialisiert auf die Gegebenheiten des DODAGs eingestellt werden. Ist dieser zu niedrig eingestellt, wertet der 6LBR unter Umständen natürliche Inkonsistenzen als Angriffe. Bei zu hoch eingestelltem Schwellwert, werden echte Angriffe potenziell nicht als solche erkannt. Ein als potenzieller Angreifer vom 6LBR identifizierter Knoten, wird daraufhin aus einer von ihm geführten *whitelist* ausgetragen. Diese *whitelist* führt alle unverdächtigen Knoten des DODAGs. In durchgeführten Simulationen erkennt das Verfahren zuverlässig *Sinkhole*s und Inkonsistenzen.
In ihren Untersuchungen stellen sie Autoren fest, dass ein DODAG der durch einen 6LBR die Kommunikation zwischen Knoten und Teilnehmern anderer Netze ermöglicht, anfällig für Angriffe aus externen Netzen ist. Zum Schutz der Knoten vor Angriffen aus fremden Netzwerken schlagen die Autoren eine dynamische Firewall zum Schutz der Knoten vor. Sie wird im 6LBR platziert und von den Knoten im DODAG individuell konfiguriert. Ein Knoten kann dabei in der Firewall auf dem 6LBR einstellen, ob an ihn gerichtete Anfragen aus anderen Netzen blockiert oder blockierte Anfragen wieder freigegeben werden sollen.
In [49] formulieren die Autoren einen Angriff der Mithilfe von Manipulation der RPL Mechanismen zur Erkennung von Inkonsistenzen im DODAG durchgeführt wird. Bei dem beschriebenen Angriff manipuliert der Angreifer Kontrollbits im Paketkopf von Datenpaketen, mit denen lokalen Inkonsistenzen in den Pfaden des DODAGs, von Knoten festgestellt werden können. Für einen konkreten Angriff, versendet der Angreifer ein Daten-Paket, dass an einen Knoten *downward* adressiert ist *upward*. Dabei setzt er die angezeigte Richtung in den Kontrollbits des *Hop-by-Hop Headers* so, als würde er erwarten, dass das Paket *upward* weitergeleitet werden soll. Der Angriff bewirkt, dass ein bis zu 2 Hops entfernter Knoten im *upward*-Pfad des Angreifers das Daten-Paket verwirft. Der beschriebene Angriff zwingt damit einen Knoten Pakete für ihn zu verwerfen, was zu einem *Sinkhole* in der Topologie führt. Da der ausgenutzte Knoten aufgrund einer Inkonsistenz die Datenpakete verwirft, initiiert er gleichzeitig einen lokalen Reparaturvorgang, indem er seinen *trickle-timer* zurücksetzt. Die Autoren schlagen als Gegenmaßnahme für den Angriff ein Verfahren vor, bei dem lokale Reparaturen aufgrund von erkannten Inkonsistenzen bewertet werden. Verursacht ein eingetroffenes Daten-Paket eine lokale Inkonsistenz, wird ein Zähler für den Versender der Nachricht erhöht. Erreicht der Zähler für einen Knoten einen statisch festgelegten Schwellwert, wird der er für eine festgelegte Zeitspanne vom Empfänger des Datenpakets ignoriert. Das Verfahren greifen die Autoren in [50] auf und erweitern den statisch gewählten Schwellwert, durch einen adaptiven Schwellwert der sich bezüglich der Position und dem Kommunikationsaufkommen jedes einzelnen Knotens im DODAG anpasst. Die Autoren simulieren ihr vorgeschlagenes Verfahren und zeigen, dass der adaptive Schwellwert die Auswirkungen des Angriffs sowie die von natürlich auftretenden Inkonsistenzen, wesentlich abmildern kann.

5 Problemanalyse der Angreifbarkeit von RPL

Die in Kapitel 4.5 vorgestellten Angriffe nutzen spezifische Schwachstellen von RPL aus, um unerlaubte Einflussnahme auf die Topologie auszuüben oder den Nachrichtenaustausch zum Vorteil eines Angreifers zu manipulieren. Die dabei ausgenutzten Schwachstellen finden sich in z.T. unterschiedlichen Angriffen wieder. In diesem Kapitel werden die Angriffe auf spezifische Parameter von RPL untersucht. Zuerst wird die Aufgabe des angegriffenen Parameters erläutert und seine Rolle im Ablauf des RPL Protokolls beschrieben. Darauf folgend wird die Anforderung für einen Angreifer formuliert, mit der es ihm möglich wird, Einfluss auf den Parameter des Knotens zu nehmen. Abschließend werden jeweils konkrete Angriffe und Auswirkungen durch den Einfluss auf den Parameter diskutiert.

5.1 Angriffe mithilfe der DODAG Versionsnummer

Ein DODAG wird eindeutig durch ein Tupel aus `RPLInstanceID` und DODAG`ID` identifiziert. Die `RPLInstanceID` ist eine einzigartige Identifikationsnummer einer RPL Instanz innerhalb eines Netzwerks. Die DODAG`ID` ist eine beliebige, jedoch routbare, IPv6-Adresse in der Topologie, die den Root eines DODAGs identifiziert und über die er erreichbar ist. Zusammen mit der DODAG`VersionNumber` beschreiben die 3 Werte eine eindeutige Iteration eines DODAGs, die als DODAG-Version bezeichnet ist. Die DODAG`VersionNumber` wird durch einen 8-Bit Sequenz-Zähler [51] repräsentiert, der ein spezifisches Alter des DODAGs angibt. Der Zähler ist streng monoton steigend und darf nur vom Root des DODAGs erhöht werden. Das Erhöhen der DODAG Versionsnummer, dient dem Root als Mittel den DODAG neu aufzubauen. Durch das Inkrementieren der DODAG-Versionsnummer, signalisiert er allen Knoten im DODAG ihre Elternknoten zu verwerfen und neue Elternknoten zur Kommunikation mit dem Root zu suchen. Bei dem Versionswechsel ersetzt ein Knoten alle seine Eltern aus der bisherigen DODAG-Version durch neue Eltern, die bereits in der inkrementierten DODAG-Version sind. Dies bewirkt, dass die Beziehungen zwischen den Knoten aufgefrischt, und der DODAG sukzessive vom Root aus in Richtung der Blätter neu aufgebaut wird. Jeder Knoten speichert die aktuelle Iteration des DODAGs, in der er sich befindet, als Tupel der `RPLInstanceID`, DODAG`ID` und der DODAG`VersionNumber`. Bei einem Versionswechsel aktualisiert der Knoten die DODAG`VersionNumber` des Tupels. Die DODAG`VersionNumber` der aktuellen Iteration eines Knotens wird von ihm in Kontrollnachrichten eingetragen, die er an seine Nachbarn verschickt, vgl. Abb. 5.1 Feld `Version Number`. Die eingetragene DODAG-Versionsnummer in DIOs, erlaubt einem Knoten zu ermitteln, ob die Iteration, in der er sich aktuell befindet, veraltet ist und inkrementiert werden muss. Mit dieser Information kann ein

Knoten entscheiden, ob die Informationen, in der empfangenen DIO aktuell sind und von ihm verarbeitet werden, oder er die Informationen verwirft und dem Versender der DIO die aktuelle DODAG-Versionsnummer mitteilt.
Lernt ein Knoten durch eine DIO, dass er sich in einer veralteten DODAG-Version befindet, migriert er in die neuere annoncierte DODAG-Version, da die Iteration, in der er sich derzeit befindet, veraltet ist. Der Wechsel in eine neue DODAG-Version ist monoton steigend und unumkehrbar. Zum vollständigen Migrieren, muss ein Knoten alle seine Eltern aus der bisherigen DODAG-Version verwerfen und ausschließlich Eltern aus der neuen DODAG-Version finden und nutzen. Solange der Knoten nicht vollständig migriert ist, trägt er dabei seine noch veraltete DODAG-Versionsnummer in versendeten Kontrollnachrichten ein. Informationen aus DIO-Kontrollnachrichten veralteter DODAG-Versionen, werden von Knoten, die bereits in eine neuere DODAG-Version migriert sind, ignoriert. Ist der Versionswechsel eines Knotens vollzogen, annonciert er seine neue Position in der aktuellen DODAG-Version an seine Nachbarn. Jeder Knoten des DODAGs beteiligt sich an dem Versionswechsel in die inkrementierte DODAG-Version. Der Versionswechsel pflanzt sich so im gesamten DODAG fort, bis letztendlich jeder Knoten in die neue DODAG Version migriert ist. Durch dieses Verfahren wird eine Iteration des DODAGs in der Topologie mit allen Knoten synchronisiert.

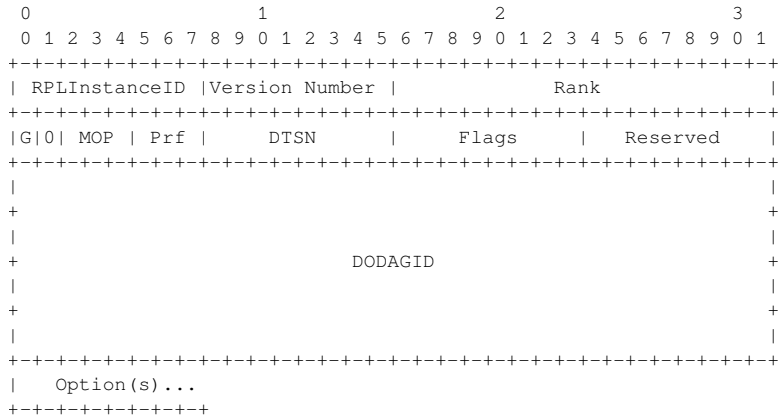

Abbildung 5.1: DIO-Kontrollnachricht

5.1.1 Voraussetzungen für Angriffe gegen die DODAG Versionsnummer

Der Angreifer muss in Kommunikationsreichweite zu den Knoten eines DODAGs sein. Er muss die gleiche Übertragungstechnik einsetzen wie die RPL-Knoten im DODAG. Er muss DIO-Kontrollnachrichten generieren und diese als ICMPv6 Nachrichten an andere Knoten versenden können. Er muss DIOs empfangen und verarbeiten können, um die Tupel der DODAG-Version für einen Angriff zu lernen. Werden gesicherte Versionen der RPL-Kontrollnachrichten im DODAG eingesetzt, muss der Angreifer Zugang zu Schlüsseln für die Kommunikation mit seinen Nachbarknoten auf Schicht 2, sowie Schlüssel zum Lesen und Erzeugen der gesicherten Kontrollnachrichten auf Schicht 3 haben. Hat ein Angreifer diese Zugangsvoraussetzungen,

kann er im DODAG teilnehmen und mit seinen RPL-Nachbarknoten Kontrollnachrichten austauschen.
Die DODAG-Versionsnummer einer DIO-Kontrollnachricht kann sowohl von RPL-Hosts als auch von RPL-Routern eingetragen werden. Damit benötigt ein Angreifer ausschließlich die Privilegien eines RPL-Hosts im DODAG, um Angriffe durchzuführen. Wird die spezifizierte Variante 0 zur gesicherten Kommunikation in der Kontrollebene des DODAGs eingesetzt, werden für einen Angriff die initial installierten oder beim Bootstrapping verteilten Schlüssel benötigt. Werden keine gesicherten Kontrollnachrichten eingesetzt, benötigt der Angreifer keine Zugangsvoraussetzungen für den Nachrichtenaustausch in der Kontrollebene.

5.1.2 Konkrete Angriffe

Ein Knoten wertet ausschließlich das `Version Number`-Feld einer empfangenen DIO-Kontrollnachricht aus, um seine Iteration des DODAGs mit der annoncierten Version einer empfangenen DIO-Kontrollnachricht abzugleichen. Die anderen Felder der DIO beinhalten keine zusätzlichen Informationen mit der ein Knoten die eingetragene `Version Number` verifizieren, und ein Inkrementieren der DODAG-Version legitimieren kann. Ein Angreifer, der Mitglied im DODAG ist, kann beliebig das `Version Number`-Feld versendeter DIO-Kontrollnachrichten verfälschen.

Pfade im DODAG auf den Angreifer ausrichten Der Angreifer nimmt als Host im DODAG teil und lernt durch empfangene DIO-Kontrollnachrichten seiner Nachbarknoten die aktuelle Version des DODAGs. Er ermittelt die nächste DODAG-Versionsnummer, indem er die den Sequenzzähler für die gelernte DODAG-Version inkrementiert. Die unerlaubt erhöhte DODAG-Versionsnummer annonciert er in DIO-Kontrollnachrichten an seine Nachbarknoten. Bei dem unerlaubten Versionswechsel ist der Angreifer zu Beginn der einzige verfügbare Knoten in der inkrementierten DODAG-Version, der als Elternknoten zur Verfügung steht. Beim Empfang der manipulierten DIO wird ein Knoten aufgrund der verfälschten DODAG-Versionsnummer feststellen, dass er sich in einer veralteten DODAG Iteration befindet. Er wird beginnen in die aktuellere DODAG Version zu migrieren und damit seine aktuellen Elternknoten zu verwerfen. Anschließend sucht er neue Elternknoten, die sich bereits in der erhöhten DODAG Version befinden. Die Nachbarknoten des Angreifers werden ihn als ihren Elternknoten wählen, da kein anderer Knoten zur Verfügung steht. Ein vom Angreifer getäuschter Knoten berechnet danach seinen neuen Rang in Abhängigkeit des annoncierten Rangs aus der verfälschten DIO und reiht sich unterhalb dem Angreifer im DODAG ein. Daraufhin annonciert er seinerseits den Versionswechsel in DIOs und veranlasst seine Nachbarknoten, die sich noch in der alten DODAG Version befinden, in die neue DODAG Version zu migrieren. Dabei reihen sich die Kinder unterhalb der getäuschten Elternknoten ein. Die Umstrukturierung pflanzt sich aufgrund der manipulierten DODAG Version in der gesamten Topologie fort, sodass letztendlich alle Pfade im DODAG auf den Angreifer ausgerichtet sind. Für den Angriff kann der Angreifer einen beliebigen gültigen Rang wählen, den er in seinen DIOs annonciert. Alle Knoten, die in seine manipulierte DODAG Version migrieren, nutzen einen Rang, der immer größer ist als der des Angreifers. Die einzige Ausnahme bildet hierbei der Root des DODAGs. Er hat als einziger

Knoten einen vorab festgelegten statischen Rang und hat niemals einen Elternknoten. Solange der Root in der veralteten DODAG Version verweilt, ignorieren alle migrierten Knoten seine veralten DIOs.

Stören der Weiterleitung von Datenpaketen im DODAG Der Angreifer richtet die Pfade der Knoten im DODAG, wie zuvor beschrieben, auf sich aus und nimmt eine zentrale Position im DODAG ein. Alle Pfade sind auf den Angreifer ausgerichtet, der damit eine Engstelle im DODAG darstellt. Datenpakete, die ein Knoten zum Root verschickt werden daraufhin immer zum Angreifer geleitet. Wenn der Angreifer in Kommunikationsreichweite zu Root ist, kann der Datenpakete selektiv an ihn weiterleiten, oder verwerfen.
Leitet der Root einen legitimen Versionswechsel ein mit dem die manipulierte DODAG-Version abgelöst wird, richten die Knoten im DODAG ihre Pfade zurück auf ihn aus. Die Neuausrichtung der Pfade im DODAG benötigt abhängig von der Tiefe des Baumes eine gewisse Zeit, bis sie zu den Blattknoten propagiert. Der Angreifer kann dann, nachdem die Neuausrichtung auf den Root vollzogen wurde, erneut den DODAG durch unerlaubtes Inkrementieren der DODAG-Version auf sich ausrichten. Durch die periodisch verursachten Neuausrichtungen der Pfade können keine langlebigen und verlässlichen Pfade zur Weiterleitung von Datenpaketen zwischen den Knoten und dem Root etabliert werden.
Hat der Angreifer die Absicht in der zentralen Position aufgrund seiner Täuschung zu verharren, kann periodisch die DODAG-Version anheben, sodass die Knoten ihre Pfade im DODAG auf ihn ausgerichtet beibehalten.

Degenerierung des DODAGs begünstigen Der Angreifer richtet die Pfade der Knoten im DODAG, wie zuvor beschrieben, auf sich aus. Dadurch nimmt er eine zentrale Position im DODAG ein. Wechselt nun der Root in die manipulierte DODAG-Version, bietet er mit versendeten DIOs seinen Nachbarknoten an, ihn als Elternknoten zu wählen. Annonciert er einen attraktiveren Rang als die gewählten Elternknoten seiner Nachbarn, wählen sie ihn als neuen Elternknoten. Die Verbesserung des Rangs der Nachbarknoten vom Root propagiert so lange in ihre *downward* Zweige, bis Knoten einen unattraktiveren Rang gegenüber ihren getäuschten Nachbarknoten in DIOs annoncieren. Da dabei der DODAG nicht von Grund auf rekonstruiert wird, degeneriert der DODAG, da teils Pfade auf den Root ausgerichtet sind, jedoch auch Pfade auf den Angreifer ausgerichtet bleiben. Die Täuschung der Knoten im DODAG hat so lange bestand, bis der Root eine neuere DODAG-Version in seinen DIOs annonciert und dadurch den DODAG neu aufbaut.

Aufbrauchen der Ressourcen aller Knoten im DODAG Die Restrukturierung des DODAGs mithilfe des unerlaubten Anhebens der DODAG Versionsnummer, propagiert vom Angreifer aus bis zu den entferntesten Knoten im DODAG. Dabei verwirft jeder Knoten seine bisherigen Elternknoten, sucht neue Eltern in der neuen DODAG Version und annonciert seine eigene Bereitschaft an die Nachbarknoten als Elternknoten genutzt zu werden. Dabei werden DIO-Kontrollnachrichten zwischen den Knoten ausgetauscht und von ihnen ausgewertet. Sind die *upward* Pfade eines Knotens durch Eintragen der Elternknoten gewählt,

annonciert er optionale *downward* Pfade an sie. Unabhängig vom MOP, versenden die Knoten DAO-Kontrollnachrichten an jeden Elternknoten, um die erreichbaren *downward* Präfixe zu annoncieren. Erst wenn alle Knoten des DODAGs migriert sind, flacht der Nachrichtenaustausch zwischen den Knoten in der Kontrollebene ab. Durch den Angriff werden die Pfade vom Root Knoten weg zum Angreifer gelenkt. Wenn der Root in die verfälschte erhöhte Version wechselt oder eine neuere DODAG Version annonciert, richten sich die Pfade zurück auf den Root aus, und verursachen erneut einen erhöhten Nachrichtenaustausch in der Kontrollebene. Der Angreifer wiederholt nun seinen Angriff indem er, nachdem die Pfade zurück auf den Root ausgerichtet sind, erneut unerlaubt die DODAG Version anhebt. Durch diese periodische Restrukturierung und dem damit verbundenen Kommunikationsaufkommen zwischen den Knoten werden die Energieressourcen der Knoten beansprucht. Zusätzlich wird dabei die verfügbare Rechenleistung sowie die Speichernutzung jedes Knotens für die Auswertung empfangener und die Vorbereitung zu versendender Kontrollnachrichten angehoben.

5.2 Angriffe mithilfe des Ranges eines Knotens

Der Rang eines Knotens ist ein abstrakter numerischer Wert, mit der die gewichtete Anzahl Hops als topologische Distanz des Knotens zum Root innerhalb eines DODAGs beschrieben wird. Der Rang wird durch einen `unsigned` 16-Bit Wert repräsentiert, der durch die im DODAG eingesetzte OF von einem Knoten ermittelt wird. In der OF0 entspricht der Rang einem `1..n`-Multiplikator von `MinHopRankIncrease`. Dabei ist der Multiplikator von 1 für den Root vorgesehen und steigt streng monoton für einen Knoten mit der Anzahl Hops, über die er den Root erreicht. Der Rang eines Knotens wird durch verschiedene Bedingungen neu berechnet. Mittels OF bestimmt ein Knoten seinen Rang, wenn er einem DODAG beitritt, seine Position im DODAG verändert oder den DODAG verlässt. Dem DODAG kann er beitreten, wenn er eine DIO von einem Nachbarknoten empfängt, der sich als Elternknoten zur Verfügung stellt. Dabei muss der Rang des potenziellen Elternknotens, ungleich dem im DODAG festgelegten `INFINITE_RANK` sein. Mit dem Rang aus der empfangenen DIO bestimmt der Knoten mittels OF seinen eigenen Rang im DODAG. Unabhängig von der eingesetzten OF im DODAG gilt immer, dass der eigene ermittelte Rank des Knotens die Relation größer als gegenüber dem Rang eines Elternknotens erfüllt. Ist der Knoten bereits Mitglied im DODAG, kann er durch Verändern des Rangs seine Position vertikal im DODAG verändern. Durch das Erhöhen seines Rangs mittels OF kann er seine Position in *downward* Richtung im DODAG versetzen. Dies erhöht seine gewichtete topologische Distanz zum Root, wodurch er von seinen Nachbarknoten als weniger attraktiv angesehen wird, als Elternknoten genutzt zu werden. Durch eine Verschlechterung des Rangs hat er ein Mittel seine Belastung in der Paketweiterleitung als Transitknoten zum Root für seine sub-DODAGs zu drosseln, da potenziell seine bisherigen Kinder auf andere Elternknoten ausweichen und diese zur Kommunikation mit dem Root nutzen.
Ein Knoten darf seinen Rang nur verbessern, wenn er von einem Nachbarn mit niedrigerem Rang als sein aktueller *Preferred Parent* eine DIO empfängt. Mit dem attraktiveren Rang aus der DIO bestimmt der Knoten mittels OF seinen neuen Rang und ersetzt den bisherigen *Preferred*

Parent durch den attraktiver platzierten Nachbarknoten. Damit steigt der Knoten *upward* im DODAG an die Position unterhalb seines neuen Elternknotens auf. Durch diese Verbesserung des Rangs steigt seine Attraktivität gegenüber Nachbarknoten, ihn als Elternknoten zu nutzen. Bei jeder Änderung des Rangs annonciert ein Knoten seinen neu bestimmten Rang in einer DIO an alle seine Nachbarknoten. Damit wird die lokale Änderung der Pfade im DODAG angezeigt und ermöglicht den Nachbarknoten auf diese zu reagieren. Verlässt ein Knoten den DODAG kann er den `INFINITE_RANK` Wert für seinen Rang in DIOs annoncieren, um sein Ausscheiden den Nachbarknoten mitzuteilen. Damit informiert er seine Kind-Knoten, dass der Pfad zum Root über ihn ab sofort nicht mehr bedient wird und er nicht als Transitknoten genutzt werden kann.

5.2.1 Voraussetzungen für Angriffe gegen den Rang

Zunächst muss der Angreife in Kommunikationsreichweite zu einem DODAG sein. Er muss die gleiche Übertragungstechnik einsetzen wie die RPL-Knoten im DODAG. Er muss DIO-Kontrollnachrichten generieren und diese als ICMPv6 Nachrichten an andere Knoten versenden können. Der Angreifer muss die OF im DODAG kennen, um einen sinnvollen Rang generieren zu können. Der Rang einer DIO-Kontrollnachricht wird in DIOs von allen teilnehmenden Knoten unabhängig ihrer Rolle als Router oder Host eingetragen. Wird eine gesicherte Kommunikation im DODAG eingesetzt, muss der Angreifer die Zugangsberechtigungen zur Teilnahme im DODAG besitzen. Authentifizieren sich die Router im DODAG durch einen separaten Schlüssel, benötigt der Angreifer diesen Schlüssel, um für seine Nachbarknoten als potenzieller Elternknoten aufzutreten. Als Mitglied kann ein angreifender Knoten das `Rank`-Feld der DIO beliebig setzen.

5.2.2 Konkrete Angriffe

Das `Rank`-Feld einer DIO teilt einem Knoten mit, welche topologische Position der Versender im DODAG hat, und ob er als potenzieller Elternknoten infrage kommt. Durch den eingetragenen Rang von DIOs erhält der Knoten Informationen über die lokale Struktur des DODAGs in seiner direkten Nachbarschaft. Mithilfe der OF kann ein Knoten feststellen, ob der eingetragene Rang die Konstruktionsgrenzen einhält. Die DIO enthält jedoch keine zusätzlichen Informationen, mit denen der Knoten den eingetragenen Rang verifizieren oder legitimieren kann.

Restrukturierung der Pfade im DODAG Ein Angreifer kann einen beliebigen Wert in das `Rank`-Feld einer DIO eintragen. Dadurch ist es ihm möglich seine topologische Position im DODAG vertikal zu verändern und damit seine Nachbarknoten zu täuschen. Jede vertikale Veränderung seiner Position im DDAOG beeinflusst die Attraktivität des Angreifers und verursacht eine Zustandsänderung auf seinen Nachbarknoten.
Verschlechtert ein Angreifer seinen Rang und annonciert ihn in einer DIO, versuchen seine bisherigen Kindknoten einen der alternativen Elternknoten mit nun attraktiverem Rang aus ihrem Eltern-Set zu wählen. Hat der Kind-Knoten einen alternativen Elternknoten, kann er

ihn nutzen, ohne seinen eigenen Rang anzupassen. Er entfernt den Angreifer als Elternknoten aus seinem Eltern-Set und nutzt an seiner Stelle den anderen. Hat der Knoten über den verworfenen Angreifer *downward* Routen annonciert und bedient, muss er diese auf andere Elternknoten seines Eltern-Sets übertragen. Hierfür wählt er Eltern aus seinem Eltern-Set aus, und benachrichtigt sie mit DAO-Kontrollnachrichten darüber, dass sie nun die Verbindung zu sub-DODAGs des Knotens bedienen. Ist der Angreifer der einzige verfügbare Knoten seines Eltern-Sets, muss er ihn als Elternknoten für die Kommunikation mit dem Root beibehalten. Hierfür passt der Knoten zunächst mittels OF seinen Rang an den verschlechterten Rang des Angreifers an und annonciert ihn seinerseits in einer DIO an seine Nachbarn. Die Verschlechterung des Rangs kann dazu führen, dass seine bisherigen Kinder, zu einem anderen attraktiveren Nachbarn migrieren. Verursacht durch die Migration seiner Kinder, propagiert die Änderung der *downward* Pfade zu erreichbaren Präfixen *upward* bis zum Root. Migrierende Knoten können optional ihrem bisherigen Elternknoten mit einer DAO-Kontrollnachricht mitteilen, dass die *downward* Route nicht mehr über ihn bedient werden soll. Hierfür versendet ein Knoten eine DAO an ihren bisherigen Elternknoten, in der eine `Target Option` zum sub-DODAG eingetragen ist, deren zugehörige `Transit Information` eine Lebensdauer von 0 im `Path Lifetime`-Feld führt. Wird die Migration nicht an den bisherigen Elternknoten kommuniziert, verfällt seine eingetragene *downward*-Route letztendlich automatisch, wenn ihre Lebensdauer abgelaufen ist.
Verbessert der Angreifer seinen Rang, indem er ihn manipuliert, steigert er damit unerlaubt seine Attraktivität als Elternknoten genutzt zu werden. Er versendet DIOs mit dem niedrigeren Rang an seine Nachbarknoten und versucht sie als Kindknoten zu gewinnen. Täuscht der Angreifer erfolgreich einen Nachbarknoten, wird ihn dieser als seinen neuen Elternknoten nutzen und als Folge den eigenen Rang verbessern. Die Verbesserung des Rangs annonciert der Nachbarknoten seinerseits in DIOs an seine Nachbarn. Wie bei der Verschlechterung des Rangs müssen durch die vertikale Positionsänderung im DODAG *downward* Routen auf andere Eltern übertragen werden. Verändert der verfälschte Rang des Angreifers lokale Beziehungen zwischen Knoten seiner Umgebung, propagiert die Änderung in alle ihre Zweige in *upward* und *downward* Richtung. Der Austausch von Kontrollnachrichten findet so lange im DODAG statt, bis die Beziehungen aller Knoten in den Zweigen auf die annoncierte Veränderung des Rangs konsolidiert sind.
Abhängig der Tiefe des DODAGs und der Rangänderung durch den Angreifer, kann die Auswirkung des Angriffs auf die Restrukturierung des DODAGs und die Zuverlässigkeit der Paketzustellung Einfluss nehmen. Dabei wird die Kommunikation in der Kontroll- sowie Datenebene erheblich beeinträchtigt[33].

Aufbrauchen der Ressourcen von Nachbarknoten Oszilliert der Angreifer seinen annoncierten Rang zwischen sehr attraktiv und `INFINITE_RANK`, bringt er seine Nachbarknoten dazu, periodisch ihre Eltern-Kind-Beziehungen zu erneuern. Die getäuschten Knoten setzen dabei immer ihren *trickle-timer* zurück und tauschen daraufhin in aufsteigenden Zeitintervallen RPL-Kontrollnachrichten aus. Durch die periodische vertikale Änderung seiner Position im DODAG verhindert der Angreifer, dass das *trickle-timer* Intervall der getäuschten

Knoten zunimmt und die Zahl der ausgetauschten Kontrollnachrichten sinkt. Die Kontrollebene im Umfeld der betroffenen Knoten ist dadurch häufig belegt und zwingt die Knoten einen Teil ihrer Ressourcen zum Versenden oder Auswerten von Kontrollnachrichten durchgehend einzusetzen. Dies setzt sich so lange in den Zweigen der Nachbarknoten des Angreifers fort, bis kein weiterer Knoten des Zweigs seinen Rang als Folge der unerlaubten Rangänderung verändert.

Erzeugen eines Sinkholes Annonciert der angreifende Knoten einen attraktiven Rang in einer DIO, lockt er seine Nachbarknoten an ihn als Elternknoten zu nutzen. Die getäuschten Nachbarknoten verwerfen ihren bisherigen bevorzugten Elternknoten, ersetzen ihn durch den Angreifer und passen ihren eigenen Rang mithilfe der OF an. Daraufhin annoncieren sie ihre neue nun attraktivere Position im DODAG. Ist die Rangänderung durch die Täuschung des Angreifers hinreichend hoch, werden die getäuschten Nachbarn des Angreifers ihrerseits Nachbarknoten als Kinder gewinnen. Die Anpassung der Rangverbesserung des Angreifers propagiert im DODAG bis alle betroffenen Zweige auf die Änderung konsolidiert sind. Der Angriff hat dadurch Auswirkung auf alle Knoten im DODAG, deren bisherige Eltern einen unattraktiveren Rang haben als der annoncierte Rang der bereits angeschwindelten Knoten. Die Pfade der erfolgreich getäuschten Knoten werden damit auf den Angreifer ausgerichtet. In dieser zentralen Position kann der Angreifer Pakete seiner sub-DODAGs in *upward*-Richtung verwerfen, oder falls er Verbindung zum Root hat, selektiv weiterleiten.

5.3 Angriffe mithilfe der Kontrollbits im Options-Header von Paketen der Datenebene

Die Schleifenfreiheit von Pfaden im DODAG wird mit versendeten Paketen in der Datenebene verifiziert [52]. Hierfür werden zusätzliche Informationen in Form einer RPL-Option in den *Hop-by-Hop Header* versendeter IPv6 Paketen eingetragen, vgl. Abb. 5.2.
Das `Option Type`-Feld der RPL-Option ist auf 0x63 festgelegt[1].
Darauf folgt das `Opt Data Len`-Feld dass die Anzahl der Bytes für die RPL-Option ab dem Feld bis zum Ende der Option beinhaltet.

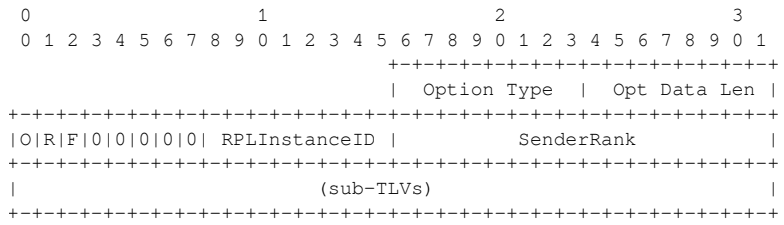

Abbildung 5.2: RPL Optionsheader

[1] http://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2

Das O-Kontrollbit des Option-Headers legt die erwartete Richtung bei jedem Hop der Paketweiterleitung fest. Wird dieses Kontrollbit gesetzt, ist die erwartete Richtung in der das Paket weitergeleitet werden soll *upward*, sonst *downward*.
Mit dem R-Kontrollbit wird angezeigt, dass das Paket zuvor einmal in entgegengesetzter Richtung zu der im O-Kontrollbit erwarteten Richtung weitergeleitet wurde.
Das F-Kontrollbit zeigt an, dass ein Knoten keinen Pfad zum Ziel des Paketes kennt.
Im `RPLInstanceID`-Feld trägt ein Knoten eine Identifikationsnummer seiner RPL Instanz ein und im darauf folgenden `senderRank`-Feld seinen aktuellen Rang.
Das `sub-TLVs`-Feld ist für mögliche zukünftige Optionen reserviert.
Die Informationen der RPL-Option dokumentieren, ob das Datenpaket bisher ohne Verletzung einer vorgegebenen Richtung weitergeleitet wurde. Ein Knoten, der das Datenpaket weiterleitet, setzt das O-Kontrollbit entsprechend der Richtung von ihm aus, in der er das Zielpräfix erreichen kann. Ist das Zielpräfix im sub-DODAG des Knotens erreichbar, setzt er das O-Kontrollbit auf *downward*. Erreicht der Knoten den Zielpräfix über einen Elternknoten, setzt er es auf *upward*. Kennt der Knoten keinen Pfad zum gegebenen Präfix des Datenpakets, verwirft er es stillschweigend im *Non-Storing MOP*. Im *Storing MOP*, kann ein Knoten das F-Kontrollbit setzten und das Paket an den Urheber des Pakets zurückschicken, falls er keinen Pfad in *downward* Richtung kennt, auf dem er das Paket zum Zielpräfix weiterleiten kann. Zum Validieren der Schleifenfreiheit bei der Paketweiterleitung von Datenpaketen im DODAG wertet ein Knoten die RPL-Option eines empfangenen Datenpakets aus. Mit dem eingetragenen Rang der RPL-Option stellt er zunächst fest, aus welcher Richtung das Paket an ihn weitergeleitet wurde und vergleicht sie mit der im O-Kontrollbit festgelegten erwarteten Richtung. Erkennt der Knoten darin eine Inkonsistenz in der Weiterleitung, setzt er das R-Kontrollbit, bevor er seinerseits die Kontrollbits für den nächsten Hop im RPL-Optionsheader einstellt und das Paket weiterleitet. Ist das Kontrollbit bereits gesetzt, verwirft er das Paket, setzt seinen *trickle-timer* zurück und startet eine lokale Reparatur im DODAG. Die Nachbarn des Knotens werden durch das Versenden von DIOs für die lokale Reparatur aufgeweckt und beteiligen sich an dem Vorgang. Im *Storing MOP* können die Knoten durch das F-Kontrollbit angezeigte Inkonsistenzen in *downward* Richtung durch einen inkrementierten Zählerstand festhalten. Erreicht dieser einen festgelegten Schwellwert von `MAX_RPL_OPTION_FORWARD_ERRORS` innerhalb einer Stunde, startet ein Knoten keine weitere lokale Reparatur, wenn erneut eine Inkonsistenz auftritt, sondern verwirft stillschweigend die inkonsistent propagierenden Datenpakete.

5.3.1 Voraussetzungen für Angriffe gegen die Kontrollbits von Datenpaketen

Der Angreifer muss in Kommunikationsreichweite zu einem DODAG sein. Er muss die gleiche Übertragungstechnik einsetzen wie die RPL-Knoten im DODAG. Wird zur Kommunikation zwischen den RPL-Knoten eine Verschlüsselung auf der Schicht 2 eingesetzt, benötigt der Angreifer die dafür eingesetzten Schlüssel.

5.3.2 Konkrete Angriffe

RPL-Optionen werden ungeschützt in IPv6 Hop-by-Hop Datenpaketen transportiert. Ein Angreifer kann aus mitgehörten Datenpaketen die `RPLInstanceID` und die Ränge seiner Nachbarknoten lernen. Diese Schwachstelle ermöglicht einem Angreifer Einfluss auf den DODAG zu nehmen, ohne die Zugangsbefugnis für die Teilnahme im DODAG zu besitzen.

Downward Routen zu einem Präfix blockieren Bei diesem Angriff muss der Angreifer ein teilnehmender Knoten im DODAGs sein. Im Storing MOP setzt der Angreifer das F-Kontrollbit der RPL-Option eines Datenpakets, dass *downward* zu einem Knoten propagiert und schickt es zurück an den Elternknoten, von dem er es erhalten hat. Damit signalisiert der Angreifer, dass er keine Route zum Empfänger des Datenpakets kennt. Hat der Elternknoten weitere Kinder mit *downward*-Routen zum Empfängerknoten, leitet er das Datenpaket über sie weiter. Hat der Elternknoten keine alternative Route, informiert er seinerseits seine Eltern darüber, dass er den Präfix des Empfängerknotens nicht erreichen kann. Hierfür setzt er das F-Kontrollbit im Options-Header des Datenpakets und leitet das Paket an sie weiter. Das Anzeigen der abgebrochenen Route propagiert so lange upward, bis ein Knoten auf dem Pfad eine alternative Route kennt, oder bis die Information beim Root eintrifft. Alle Knoten, die keinen alternativen Pfad zum Präfix kennen, löschen ihn aus ihren *downward* erreichbaren Präfixen. Dadurch erreicht der Angreifer, dass Pakete auf den erfolgreich getäuschten Knoten im *upward* Pfad, nicht mehr an das blockierte Präfix *downward* weitergeleitet werden. Kann der Angreifer genügend Nachbarknoten täuschen, sodass alle existierenden Pfade zum Präfix gelöscht werden, erreicht er, dass ein Sinkhole für das Präfix im DODAG entsteht. Im speziellen Fall, dass das blockierte Präfix *downward* nur über den Angreifer erreichbar ist, blockiert der Angriff direkt alle *downward* Routen zum blockierten Präfix oberhalb des Angreifers. Durch die Position an der Engstelle hat der Angreifer ohnehin schon die triviale Möglichkeit direkt alle Pakete zum blockierten Präfix selektiv zu verwerfen. Das Verwerfen von Paketen kann jedoch unter Umständen von den Nachbarknoten aufgedeckt werden.

Lokale Reparatur erzwingen Der Angreifer kreiert zunächst ein IPv6 Datenpaket, in das er eine zuvor mitgehörte RPL-Option und den Inhalt eines abgefangenen Datenpakets kopiert. Durch die Kopie der RPL-Option trägt der Angreifer die `RPLInstanceID` sowie einen gültigen Rang eines Knotens. Durch das O-Kontrollbit der mitgehörten RPL-Option lernt der Angreifer die Beziehung zwischen dem Versender und dem Empfänger des abgefangenen Datenpakets. Nun kann der Angreifer durch Manipulieren der Kontrollbits der kopierten RPL-Option eine Inkonsistenz herbeiführen. Kippt der Angreifer das O-Kontrollbit der ursprünglichen RPL-Option, ändert er die erwartete Richtung, auf der das Datenpaket ausgehend von dem eingetragenen Rang propagieren soll. Zusätzlich setzt er das R-Kontrollbit, um anzuzeigen, dass das Paket bereits zuvor einmal in die entgegengesetzte Richtung zur Erwarteten propagiert ist. Das manipulierte Datenpaket schickt er an den ursprünglichen Empfänger. Beim Auswerten der Kontrollbits im Hop-by-Hop Header stellt der Empfänger fest, dass das Paket in die zum O-Kontrollbit entgegengesetzte Richtung weitergeleitet wurde. Da das R-Kontrollbit bereits gesetzt ist, wertet dies der Knoten als eine Inkonsistenz. Daraufhin verwirft er das Datenpaket

und initiiert einen lokalen Reparaturvorgang. Hierfür setzt er seinen *trickle-timer* zurück und versendet DIO-Kontrollnachrichten auf der Multicast-Adresse an seine Nachbarknoten.

Sinkhole für Datenpakete bei einem Nachbarknoten erzeugen Bei diesem Angriff muss der Angreifer ein Mitglied im DODAGs sein. Der Angreifer kreiert zunächst ein IPv6 Datenpaket, in das er eine zuvor mitgehörte RPL-Option und den Inhalt des Pakets kopiert. Er setzt das R-Kontrollbit, mit dem er anzeigt, dass das Paket schon zuvor in entgegengesetzter Richtung zum O-Kontrollbit weitergeleitet wurde. Das Datenpaket wird dann vom Angreifer an einen Nachbarn weitergeleitet und das O-Kontrollbit entgegengesetzt der erwarteten Richtung, in der es verschickt wird gesetzt. Der Nachbarknoten des Angreifers verwirft daraufhin das empfangene Datenpaket und startet eine lokale Reparatur. Setzt der Angreifer die Kontrollbits O und R für alle Datenpakete so, dass seine Nachbarn diese verwerfen, zwingt er sie dazu für ihn ein Sinkhole zu generieren. Im *Storing MOP* kann der Angriff *downward* ausgeweitet werden, wenn die Knoten aufgetretene Inkonsistenzen mitzählen. Forciert der Angreifer innerhalb einer Stunde eine Anzahl von reparierten Inkonsistenzen die `MAX_RPL_OPTION_FORWARD_ERRORS` übersteigt, hört der angegriffene Knoten auf weitere Reparaturen bei festgestellten Inkonsistenzen einzuleiten und verwirft stattdessen immer die Daten-Pakete. Dadurch bringt ein Angreifer den angegriffenen Knoten dazu, ein *Sinkhole* für ihn zu erzeugen und verschleiert damit der Urheber des Angriffs zu sein.

5.4 Angriffe mithilfe von DIS-Kontrollnachrichten

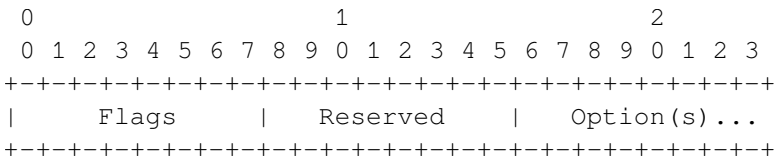

Abbildung 5.3: DIS-Kontrollnachricht

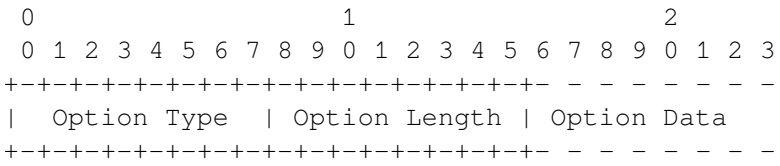

Abbildung 5.4: Option Felder

DIS-Kontrollnachrichten werden eingesetzt, um DIO Nachrichten von Nachbarknoten anzufordern. Eine DIS-Kontrollnachricht beginnt mit einem für zukünftige Einstellungen reservierten 8-Bit `Flags`-Feld, gefolgt von einem ungenutzten 8-Bit `Reserved`-Feld, vgl. Abb. 5.3. Optional können dahinter weitere Informationen an die DIS in einer variablen Anzahl `Option(s)` angehängt werden. Eine Option beginnt mit einem 8-Bit `Option Type`-Feld,

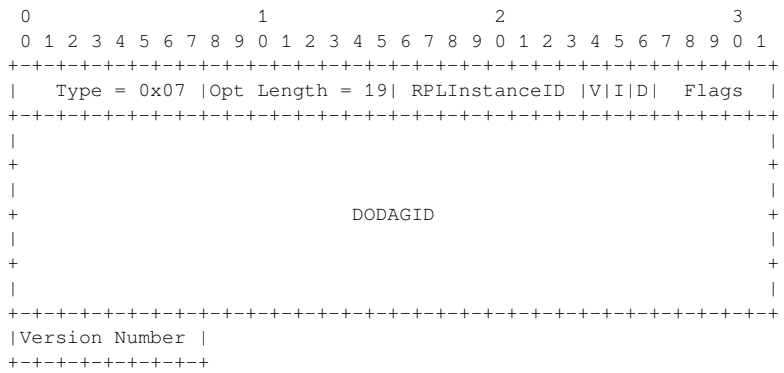

Abbildung 5.5: DIS Solicited Information Option

in dem ihr Einsatzgebiet codiert ist. Darauf folgt ein `Option Length` Feld, in dem die Anzahl der Bytes ab dem Feld bis zum Ende der Option angegeben ist, vgl. Abb. 5.4. In den folgenden Bytes der `Option Data` befinden sich die konkreten Informationen der Option. Eine Ausnahme ist die `Pad1` Option, die mit dem Typ 0x00 identifiziert ist. Sie besteht nur aus dem `Option Type`-Feld um 1-Byte aufzufüllen. Die `PadN` Option (0x01) füllt die `Option Data` mit leeren Bytes auf. Die Anzahl der aufzufüllenden Bytes ist im `Option Length`-Feld angegeben. Diese beiden Optionen erlauben es, eine Kontrollnachricht auf Vorgaben der Anordnung von Daten in einer Kontrollnachricht auszurichten. Zusätzliche Informationen, die ein Empfängerknoten als Filterkriterium nutzt, können in einer `Solicited Information` Option eingetragen werden, vgl. Abb. 5.5. Dieser Optionstyp wird mit 0x07 im `Option Type`-Feld gekennzeichnet. Die Länge dieser Option ist mit 19 festgelegt und wird im `Option Length`-Feld eingetragen. Das `RPLIndtanceID`-Feld gibt an, für welche RPL Instanz der Knoten DIOs anfordert. Darauf folgen 3 Kontrollbits, die festlegen welche Filtermöglichkeiten der Knoten mit den nachfolgenden Informationen der DIS hat. Das V-Kontrollbit legt fest, ob die DODAG-Versionsnummer berücksichtigt werden soll. Ist das `I`-Kontrollbit gesetzt, soll die `RPLInstanceID` berücksichtigt werden. Das D-Kontrollbit gibt an, ob die eingetragene `DODAGID` zum Filtern genutzt werden soll. Sind die jeweiligen Kontrollbits nicht gesetzt, werden die zugehörigen Felder durch den Versender der DIS mit Nullen aufgefüllt und vom Empfänger ignoriert. DIS-Kontrollnachrichten können per Unicast an einen Nachbarknoten oder per Multicast an alle Nachbarknoten versandt werden. Eine als Unicast-Paket versandte DIS-Kontrollnachricht, wird mit einer per Unicast versandten DIO beantwortet. Enthält die als Unicast-Paket versandte DIS eine `Solicited Information` Option, beantwortet sie der Empfänger nur, wenn die eingetragenen Filterkriterien von ihm erfüllt werden. Enthält die DIS keine `Solicited Information` Option, beinhaltet die als Antwort per Unicast versandte DIO dann eine *DODAG Configuration Option*. Sie enthält Informationen, mit denen der Knoten lernt welche Voraussetzungen und Einstellungen benötigt werden um dem DODAG beizutreten. Eine per Multicast versandte DIS, wird von einem Empfänger als lokale Inkonsistenz im DODAG gewertet, wenn sie keine `Solicited Information` Option beinhaltet oder alle ihre eingetragenen Filterkriterien erfüllt sind.

5.4.1 Voraussetzungen für Angriffe mit DIS-Kontrollnachrichten

Der Angreifer muss in Kommunikationsreichweite zu einem DODAG sein. Er muss die gleiche Übertragungstechnik einsetzen wie die RPL-Knoten im DODAG. Wird zur Kommunikation zwischen den RPL-Knoten eine Verschlüsselung auf der Schicht 2 eingesetzt, benötigt der Angreifer die dafür eingesetzten Schlüssel.

5.4.2 Konkrete Angriffe

Mit dem Versenden von DIS-Kontrollnachrichten wird einem Angreifer ermöglicht den Energieverbrauch der Knoten seiner Umgebung zu erhöhen, indem er seine Nachbarknoten zwingt die DIS auszuwerten und darauf zu antworten. RPL versendet immer eine DIO-Kontrollnachricht auf eine empfangene DIS. Der einzige Regelmechanismus eines RPL Knotens nicht auf eine DIS zu antworten, wird durch die eingetragenen Filterkriterien der DIS festgelegt.

Stören des Übertragungsmediums und der Nachbarknoten Der Angreifer versendet periodisch DIS-Knontrollnachrichten per Multicast an alle seine Nachbarknoten. Die versendete DIS enthält keine eingetragenen Filterinformationen für die Empfänger, sodass die DIS als lokale Inkonsistenz von den Empfängerknoten gewertet wird. Die Knoten setzen daraufhin ihren *trickle-timer* zurück und beginnen DIO-Kontrollnachrichten auszutauschen. Wählt der Angreifer ein kurzes Zeitintervall zwischen dem Versenden der DIS-Kontrollnachrichten, wird das Übertragungsmedium häufig mit DIO-Kontrollnachrichten belegt. Während des Versendens sowie beim Verarbeiten der DIOs, werden von den Knoten Ressourcen beansprucht und ihr Energieverbrauch angehoben. In diesen kurzen, aber häufigen Intervallen werden die betroffenen Knoten von ihrem ursprünglichen Einsatzziel im DODAG abgelenkt.

5.5 Angriffe mithilfe von DAO-Kontrollnachrichten

DAO-Kontrollnachrichten werden dazu eingesetzt *downward* Routen *upward* zu annoncieren. Eine DAO wird per Unicast *upward* an eine Untermenge der verfügbaren Eltern eines Knotens versendet. Im *Storing MOP* ist die Zieladresse der DAO direkt der benachrichtigte Elternknoten. Im *Non-Stroring MOP* wird die DAO, direkt an den Root des DODAGs verschickt. Eine DAO-Kontrollnachricht beginnt mit einem `RPLInstanceID`-Feld. Dort trägt der Versender ein, für welche RPL Instanz er *downward* Präfixe annonciert, vgl . Abb. 5.6. Darauf folgt das K-Kontrollbit, mit dem der Versender einer DAO, den Empfang durch eine DAO-ACK-Kontrollnachricht als Empfangsbestätigung von dem benachrichtigten Elternknoten anfordert. Das D-Kontrollbit zeigt an, ob in der DAO eine `DODAGID` eingetragen ist. Im `DAOSequence`-Feld wird eine Sequenznummer eingetragen, die vom Knoten zum Zuordnen einer DAO-ACK Empfangsbestätigung genutzt wird. Die optionale `DODAGID` wird üblicherweise nur eingetragen, wenn die DAO an einen Knoten in einer lokalen RPL Instanz versendet wird. In einer globalen RPL Instanz, ist die `DODAGID` statisch und allen teilnehmenden Knoten bekannt.

Darauf folgen Optionen in der DAO, die konkret Informationen über erreichbare Präfixe des sub-DODAGs eines Kindes, *upward* für seine Elternknoten bereitstellen. Diese beinhalten

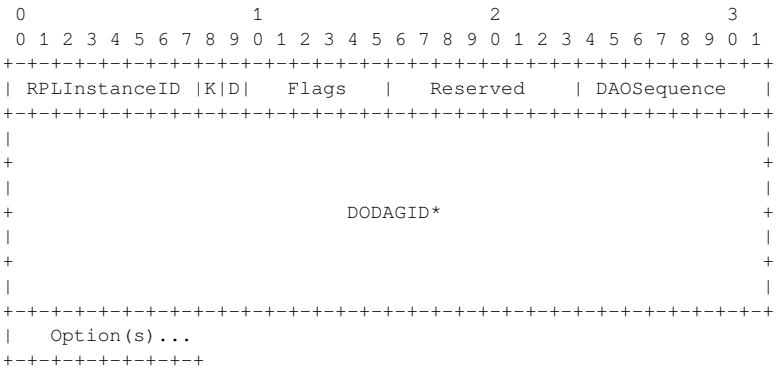

Abbildung 5.6: DAO-Kontrollnachricht

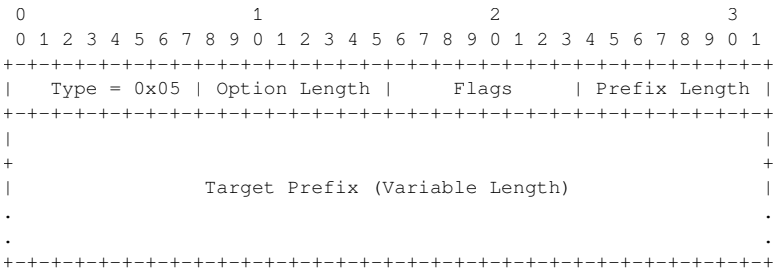

Abbildung 5.7: RPL Target Option

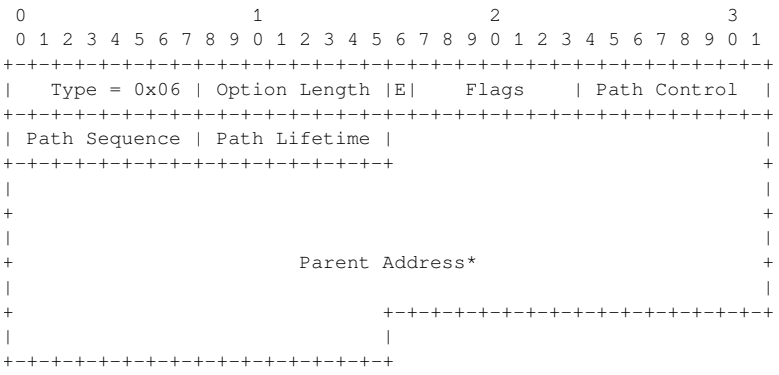

Abbildung 5.8: RPL Transit Information

die Kombination aus einer RPL `Target` Option, vgl. Abb. 5.7, auf die eine oder mehrere zugehörige RPL `Transit Information` Abschnitte folgen, vgl. Abb. 5.8. Eine RPL `Target` Option ist als `Type` 0x05 gekennzeichnet. Darauf folgt das `Option Length`-Feld, in dem die Anzahl Bytes der Option angegeben ist, gefolgt von einem reservierten `Flags`-Feld für zukünftigen Einsatz. Das `Prefix Length`-Feld gibt die Anzahl der signifikanten Bits des Präfixes an, der im darauf folgenden `Target Prefix`-Feld eingetragen wird. Das

eingetragene Präfix muss eine Multicast-, oder eine erreichbare Host-Adresse im sub-DODAG des Kindes sein. Nach der `RPL Target` Option, folgt eine oder mehrere `RPL Transit Information` Abschnitte, die mit dem Typ 0x06 gekennzeichnet sind und zur vorangestellten `RPL Target` Option gehören. Eine `RPL Transit Information` enthält Informationen, die von Knoten im *upward* Pfad für das Zusammenstellen von Source-Routen zum Versender der DAO genutzt werden. Im *Non-Storing MOP* ist die Information aus den angehängten Optionen für den Root vorgesehen. Hinter dem `Option Length`-Feld der Option, befindet sich das E-Kontrollbit mit dem angezeigt wird, ob der eingetragene Elternknoten in der `RPL Transit Information` externe Präfixe mit dem RPL-Netzwerk verbindet. Ein externes Präfix den der Knoten erreicht, ist dann in der vorangestellten `RPL Target` Option eingetragen. Das Kontrollbit-Feld `Flags` ist ungenutzt und für den zukünftigen Einsatz reserviert. Mit dem darauf folgenden `Path Control`-Feld, kann der Knoten seine Präferenz für einen Elternknoten indizieren, über den er bevorzugt mit Nachrichten *downward* versorgt werden möchte. Die angezeigte Präferenz des Elternknotens sinkt vom LSB zum MSB des Feldes. Die eingetragene Präferenz für einen *donward* Elternknoten kann vom Root berücksichtigt werden, wenn er *downward* mit dem Knoten kommuniziert. Das darauf folgende `Path Sequence`-Feld beinhaltet einen Zähler, der immer durch den initialen Urheber einer DAO inkrementiert wird, sobald er eine Aktualisierung der Informationen zu einem *downward* Pfad annonciert. Hierfür versendet er die aktualisierte Information in einer DAO mit einem erhöhten Zählerstand. Diese beinhaltet den ursprünglichen `Target Prefix` in der `RPL Target` Option für den *downward* Pfad sowie die inkrementierte `Path Sequence` in der zugehörigen *Transit Information*. Dahinter wird im `Path Lifetime`-Feld die Lebensdauer des annoncierten Pfades zum Präfix angegeben. Darauf folgt die Adresse des Elternknotens, wenn der DODAG im *Non-Storing MOP* betrieben wird. Im *Non-Storing MOP* werden DAOs direkt an den Root verschickt und von Knoten auf dem *upward* Pfad direkt zu ihren jeweiligen Eltern weitergereicht, bis sie zum Root propagiert. Der Root sammelt die Informationen aus den eingetroffenen DAOs der Knoten im DOODAG, und kann aus den Informationen *Source-Routen* generieren, mit denen er die Knoten erreicht. Im *Storing MOP* werden Pfade zu erreichbaren Präfixen, zusätzlich auf den Elternknoten in einer Routingtabelle gespeichert und für ein hop-by-hop Routing *downward* eingesetzt. Gelernte erreichbare Präfixe werden von den Knoten dann ihrerseits an Elternknoten in DAOs annonciert. Damit entstehen in den *upward* Pfaden der Knoten, Hop-by-Hop Routen zu *downward* erreichbaren Präfixen im DODAG.

5.5.1 Voraussetzungen für Angriffe

Der Angreifer muss in Kommunikationsreichweite zu einem DODAG sein. Er muss die gleiche Übertragungstechnik einsetzen wie die RPL-Knoten im DODAG. Der eingesetzte MOP im DODAG muss *downward* Routen erlauben. Der Angreifer muss ein Mitglied im DODAG sein.

5.5.2 Konkrete Angriffe

Ein Knoten kann in einer DAO beliebige Präfixe in einer beliebigen Anzahl an seine Eltern annoncieren. Dies beinhaltet sowohl Präfixe innerhalb des DODAGs, als auch externe Präfixe.

Ein Angreifer kann dies ausnutzen, um fiktive Knoten in den DODAG annoncieren. Er kann alternative Pfade zu *downward* Präfixen erzeugen, die das annoncierte Präfix nicht erreichen können. Zusätzlich kann er existierende Pfade zu *downward* Präfixen in seinem *upward* Pfad wegbrechen lassen.

Blockieren der *downward* Routen von Knoten Im *Storing MOP* können annoncierte Präfixe und ihre eingetragene Lebensdauer einer DAO nicht von den Elternknoten überprüft werden. Ein Knoten, der von seinem Kind einen *downward* Präfix gelernt hat, erzeugt einen Eintrag in seiner lokalen Routing Tabelle. Der Eintrag bleibt so lange erhalten, bis die Lebensdauer für den Präfix abläuft oder sein Kind eine *no Path*-DAO für das Präfix verschickt und damit die Erreichbarkeit des Präfixes abmeldet. Die eingetragene Lebensdauer für das annoncierte Präfix ist ein Multiplikator für `Lifetime Units`. Die Zeitspanne einer `Lifetime Unit` ist individuell für einen DODAG festgelegt und allen Knoten durch die DODAG `Configuration Option` bekannt. Sind die Bits des `Path Lifetime`-Felds einer DAO alle 1, gilt die Lebensdauer für das Präfix als unendlich. In diesem Fall bleibt der Eintrag bis zum Empfang einer *no Path*-DAO zum Präfix in der Routing Tabelle erhalten. Zum Blockieren der *downward* Routen im *upward* Pfad annonciert der Angreifer fiktive Präfixe mit unendlicher Lebensdauer im sub-DODAG seiner Elternknoten. Die Eltern des Angreifers erzeugen donward Einträge in ihren Routing Tabellen für die fiktiven Präfixe und annoncieren ihrerseits die gelernten Präfixe *upward* an ihre Eltern. Der Angreifer wiederholt den Vorgang bis seine Eltern keinen weiteren Eintrag in ihren Routing Tabellen für die Kommunikation *downward* aufnehmen können. Wenn die Knoten im DODAG homogene Hardware nutzen und identische Größen von flüchtigem Speicher zur Verfügung haben, werden die Routing Tabellen aller Knoten im *upward* Pfad des Angreifers mit seinen fiktiv erreichbaren Präfixen aufgefüllt. Als Resultat können keine weiteren *downward* Präfixe im erfolgreich getäuschtem *upward* Pfad gelernt werden. Wird ein neues Präfix an einen angegriffenen getäuschten Elternknoten annonciert, der keinen weiteren Eintrag in seine Routingtabelle aufnehmen kann, wird dieser keine Pakete *downward* an das neu annoncierte Präfix weiterleiten. Mit einer DAO-ACK als Antwort, kann der Elternknoten optional seinem Kind mitteilen, dass er den *downward* Pfad für das annoncierte Präfix nicht für ihn bedienen kann. Hat der Kindknoten einen alternativen Elternknoten für das Präfix, kann er versuchen *downward* über ihn erreichbar zu sein. Werden bei der Kommunikation von DAOs keine DAO-ACK Bestätigungsnachrichten ausgetauscht oder gehen verloren, hat ein Knoten keine Anhaltspunkte darüber, ob seine annoncierten Präfixe vom Elternknoten bedient werden. Als Folge eines erfolgreichen Angriffs, bleiben die blockierten Präfixe unbekannt im *upward* Pfad. Die erfolgreich getäuschten Knoten im *upward* Pfad eines blockierten Präfixes leiten keine Pakete *downward* weiter an sie. Durch die Täuschung bringt der Angreifer die Knoten dazu, alle explizit an ein blockiertes Präfix gerichtete Paketweiterleitung zu unterbinden und dadurch ein verteiltes Sinkhole gegen die Präfixe zu erzeugen. Sind bereits *downward* Präfixe in ihren *upward* Pfaden bekannt, kann der Angreifer durch fortwährendes annoncieren fiktiver Präfixe die echten Präfixe aus den Routingtabellen der Knoten verdrängen. Haben die annoncierten Präfixe von ehrlichen Knoten eine limitierte Lebensdauer, verdrängt jedes neu gelernte Präfix auf einem Knoten einen abgelaufenen alten, sobald die Routingtabelle aufgefüllt ist. Annonciert

der Angreifer fiktive Präfixe mit unendlicher oder sehr langer Lebensdauer, werden graduell alle echten erreichbaren Präfixe in der Routingtabelle, von den fiktiven Präfixen des Angreifers verdrängt.
Im *Non-Storing MOP* trägt jeder Knoten sein Präfix sowie den seines Elternknotens in die RPL `Target` Option einer DAO ein und versendet sie direkt an den Root des DODAGs. Die DAO wird von jedem Knoten unverändert an seinen Elternknoten weitergeleitet, bis sie beim Root eintrifft. Dadurch kann der Root alle Knoten, sowie deren Eltern-Kind Beziehungen, kennenlernen und daraus Source-Routen für die *downward* Kommunikation mit ihnen generieren. Kann der Angreifer, durch das annoncieren von fiktiven Präfixen, die Routing Tabelle des Roots erfolgreich auffüllen, blockiert er die Präfixe im gesamten DODAG.

Inkonsistenzen im DODAG erzeugen Werden keine oder nur symmetrische Gruppenschlüssel zur gesicherten Kommunikation eingesetzt, kann ein Angreifer eine DAO seiner Nachbarknoten mithören und auswerten. Mit den gelernten Informationen kann er eine DAO erstellen, in der er das eingetragene Präfix des Elternknotens und das Präfix des Kindes vertauscht. Im *Storing MOP* verschickt er eine so manipulierte DAO an seine Elternknoten. Sie annoncieren ihrerseits die manipulierte eingetragene Beziehung zwischen den erreichbaren Präfixen weiter in ihren *upward* Pfaden. Wird nun eine Nachricht *downward* an einen der Präfixe versendet, besteht durch die Täuschung die Chance, dass die Nachricht beim Weiterleiten zum Zielpräfix die Richtung wechselt und damit inkonsistent weitergeleitet wird. Dabei wird von den direkt in der Weiterleitung beteiligten Knoten eine lokale Inkonsistenz festgestellt, die ihren *trickle-timer* zurücksetzen und einen lokalen Reparaturvorgang einleiten.
Im *Non-Storing MOP* verschickt der Angreifer eine manipulierte DAO an den Root des DODAGs. Dieser wird beim Konstruieren der Pfade zu den Knoten eine Inkonsistenz feststellen und potenziell einen globalen Reparaturvorgang einleiten.

5.6 Angriffe mithilfe der Destination Advertisement Trigger Sequence Number in DIO-Kontrollnachrichten

Die *Destination Advertisement Trigger Sequence Number* (DTSN) ist eine Sequenznummer, die von Elternknoten in DIO Nachrichten eingetragen wird. Sie zeigt die Iteration zu *downward* Routen in seinen jeweiligen sub-DODAG an. Nimmt ein Knoten im DODAG teil, übernimmt er die DTSN aus der DIO seines Elternknotens. Die gelernte DTSN trägt er dann im `Path Sequence`-Feld der RPL `TransitInformation` von DAOs ein, die er an den Elternknoten verschickt. Erhöht der Elternknoten die DTSN in seinen DIO-Kontrollnachrichten, fordert er damit seine Kinder auf DAOs an ihn zu verschicken, mit denen er aktuelle Informationen zu erreichbaren Präfixen seiner sub-DODAGs sammelt. Im *Storing MOP* identifiziert die DTSN den Pfad zwischen einem Elternknoten und seinen direkten Kindern. Ein Kindknoten speichert zu jedem seiner Elternknoten eine DTSN. Eine erhöhte DTSN aus einer DIO der Elternknoten wird von den Kindern entsprechend übernommen und mit einer DAO beantwortet, in die das Kind seine erreichbaren *downward* Präfixe einträgt, die der Elternknoten bedienen soll. Im *Non-Storing MOP* gilt eine globale DTSN für den gesamten DODAG. Sie wird vom Root

initiiert und von den teilnehmenden Knoten in DIOs genutzt. Nur der Root darf im *Non-Storing MOP* die DTSN erhöhen. Eine inkrementierte DTSN wird von jedem Knoten des DODAGs übernommen und *downward* in DIOs weitergegeben. Die als Antwort verschickte DAO wird im *Non-Storing MOP* verzögert an den Elternknoten verschickt. Mit steigender Distanz zum Root sinkt die Verzögerung, bevor eine DAO vom Knoten verschickt wird. Damit soll die Aktualisierung der Pfade möglichst tief im DODAG beginnen und näher am Root liegenden Knoten ermöglichen viele gelernte Präfixe in einer DAO zu versenden. Dadurch soll die Anzahl der versendeten DAOs im DODAG balanciert werden.

5.6.1 Voraussetzungen für Angriffe

Der Angreifer muss in Kommunikationsreichweite zu einem DODAG sein. Er muss die gleiche Übertragungstechnik einsetzen wie die RPL-Knoten im DODAG. Der Angreifer muss Mitglied im DODAG sein. Der eingesetzte MOP im DODAG muss *downward* Routen erlauben.

5.6.2 Konkrete Angriffe

Ein Angreifer kann einen beliebigen Wert als DTSN in einer versendeten DIO eintragen. Im *Storing MOP* kann er seine Kinder fortwährend dazu bringen DAOs an ihn zu schicken. Hierfür versendet er in jeder DIO eine inkrementierte DTSN. Beim Empfang einer DIO mit inkrementierter DTSN, wird auf dem Knoten eine DAO vorbereitet, in der Knoten alle von *downward* Präfixe einträgt, die vom Angreifer bedient werden sollen. Die Kinder antworten mit der DAO und passen ihre DTSN für den Elternknoten entsprechend an.
Im *Non-Storing MOP* speichert jeder Knoten nur die DTSN vom Root. Lernt er eine neue DTSN, passt er sie entsprechen der inkrementierten DTSN aus DIOs an. Eine inkrementierte DTSN propagiert in DIOs *downward* bis zu den Blättern des DODAGs. Jeder Knoten wartet eine gewisse Zeit, bevor er seine erreichbaren Zieladressen an die Eltern weiterreicht. Im einfachsten Fall kann die Dauer in direkter Abhängigkeit zum Rang festgelegt werden. Je kleiner der Rank, umso länger wartet der Knoten, bis er eine DAO verschickt.

Flut von DAO-Kontrollnachrichten *upward* erzeugen Wird der DODAG im *Non-Storing MOP* betrieben kann der Angreifer alle Knoten des DODAGs dazu bringen, DAO-Kontrollnachrichten an den Root zu verschicken. Hierfür nimmt er zunächst als Knoten im DODAG teil und lernt die aktuell vom Root eingesetzte DTSN. Er erhöht unerlaubt die Sequenznummer und versendet sie in DIOs an seine Nachbarn. In DIOs annonciert, propagiert die manipulierte DTSN zu allen Knoten im DODAG. Jeder Knoten der die neue DTSN lernt bereitet eine DAO vor, in der jedes von ihm aus erreichbare Präfix eingetragen wird, und versenden die DAO an den Root.
Der Angreifer kann den Angriff beliebig oft wiederholen und so die Ressourcen der Knoten im DODAG für das erzeugen sowie Versenden von DAOs erschöpfen. Nutzt der Root die bisherige und nun veraltete legitime DTSN reagieren die getäuschten Knoten nicht mehr auf seine Anfrage ihm die *downward* Präfixe mitzuteilen.

6 Analyse der Verarbeitungskette von RPL in RIOT

In diesem Kapitel wird die Verarbeitung von eingehenden und ausgehenden Paketen im RIOT Netzwerkstack erläutert. Anschließend wird die Verarbeitung von DIO, DIS, DAO, DAO-ACK-Kontrollnachrichten und die von angehängten RPL Optionen im Options-Paketkopf von IPv6 Datenpaketen in der RIOT Implementierung hinsichtlich der Zustandsänderungen des Knotens analysiert. Da die RPL Implementierung von RIOT derzeit keine CC-Kontrollnachrichten unterstützt findet diesbezüglich keine Untersuchung statt.

6.1 Funktionsaufbau des RIOT Netzwerkstacks

Der RIOT Netzwerkstack ist unterteilt in die Verwaltung von Protokollebenen sowie der konkreten Implementierungen von Funktionen innerhalb der Protokollschichten. Dabei sind die Schichten voneinander getrennt und kommunizieren über das `netreg` Kommunikationsmodul miteinander. Die `netreg` beinhaltet ein Register, in dem Threads aus den Schichten der Netzwerkstacks typisiert eingetragen und den Schichten zugeordnet werden können. Jeder registrierte Thread übernimmt die Aufgabe seiner Schicht oder eines speziellen Einsatzgebiets innerhalb der Schicht. Eingehende sowie ausgehende Pakete werden im RIOT Netzwerkstack mittels Weiterleitung von *IPC-Nachrichten* zwischen den Threads der Schichten weitergereicht. Das Weiterleiten von *IPC-Nachrichten* erlaubt den Threads zudem zusätzliche Informationen mit Threads aus anderen Schichten austauschen.
Ein Thread, der die Aufgabe einer Schicht übernimmt, registriert sich während seiner Initialisierung bei der `netreg`. Für die Registrierung eines Threads wird zuerst festgelegt, welcher Nachrichtentyp beim Weiterleiten von *IPC-Nachrichten* relevante Informationen für ihn bereithält. Die `netreg` speichert dieses Filterkriterium sowie die PID des Threads, um relevante Nachrichten gezielt an ihn weiterzugeben. Das konkrete Weiterreichen von Paketen zwischen den Schichten im Netzwerkstack wird von der `netapi` übernommen. Sie implementiert Funktionen, die von den Threads der Schichten aufgerufen werden, um ein Paket zur Weiterverarbeitung an die nächste zuständige Schicht weiterzugeben. Dabei ist die Abfolge der Verteilung in Abhängigkeit des Pakettypen sowie der Empfangs- und Senderichtung in der `netapi` vorgegeben. Dadurch müssen die einzelnen Threads der Schichten keine explizite Kenntnis voneinander haben, um Nachrichten auszutauschen und die Verarbeitung eines Pakets durch die nächste Schicht fortsetzen zu lassen. Die `netapi` greift für die Kommunikation zwischen den Threads auf das Register der `netreg` zu. Die dort zu bestimmten Nachrichtentypen registrierten Threads, werden benachrichtigt sobald die `netapi` ein Paket zum Weiterverarbeiten verteilen soll. Durch die Zuordnung der registrierten Threads zu ihrem

Interesse an bestimmten Nachrichtentypen kann die netapi die Weiterleitung von Paketen an die jeweils in der Verarbeitungsrichtung nächste verantwortliche Schicht weitergeben. Als Verwaltungsmodul bietet die netreg zudem Funktionen zum Registrieren von Threads zu einem bestimmten Nachrichtentyp und zum Abmelden von Threads aus seinem Register.
Pakete werden im RIOT Netzwerkstack durch eine einfach verkettete Liste von pktsnip Datenstrukturen gekapselt und weitergereicht. Sie repräsentieren einen Ausschnitt eines Pakets mit jeweils vorangestellten Verwaltungs- und Metainformationen. Die pktsnips werden mit Hilfsfunktionen des RIOT pktbuffers erstellt, verkettet und verwaltet. Jeder pktsnip ist dabei während seiner Lebensdauer in einem zentralen Puffer des pktbuffer für die Verarbeitung zwischengespeichert. In einem pktsnip sind die Metainformationen in Feldern unterteilt, die den Inhalt der transportierten Daten des Paketausschnitts beschreiben. Jeder pktsnip hat einen festgelegten Typ, der seine zugehörige Schicht im Netzwerkstack angibt. Er hat einen Benutzerzähler, mit dem die Anzahl aktiver Nutzer eingetragen wird, um bei konkurrierenden Schreibzugriffen ein *copy-on-write* auszuführen und den pktsnip zu kopieren. Ein pktsnip hat einen Zeiger auf den jeweils nachfolgenden pktnip der den nächsten Ausschnitt des Pakets darstellt, bspw. der hat ein pktsnip eines IPv6 Paketkopfs einen Zeiger auf den pktsnip mit einem UDP Paketkopf im Paket. Hinter den Metainformationen beinhaltet der pktsnip einen Datenbereich, in dem die eingetragenen Rohdaten der jeweiligen Schicht eines Pakets abgelegt sind, bspw. die konkreten Daten des IPv6 Paketkopfs aus der Sicherungsschicht.

6.1.1 Verarbeitung beim Empfang eines Pakets

Wird ein Paket durch einen RIOT Knoten empfangen, verarbeitet der konkrete Treiber des Transceivers die Daten und leitet diese an das gnrc_netdev2 Modul. In dem gnrc_netdev2 Modul wird ein einheitlicher MAC-Paketkopf für das empfangene Paket erstellt und ihm vorangestellt. Der vorangestellte einheitliche MAC-Paketkopf mit dem dahinterliegenden empfangenen Paket, wird in einem typisierten pktsnip mittels netapi an den Netzwerkstack von RIOT weitergereicht. Dadurch bildet das gnrc_netdev2 Modul die erste Abstraktionsebene im RIOT Netzwerkstack für eingehende Pakete. Zur Weiterverarbeitung wird die netreg von der netapi nach allen Threads durchsucht, die sich für den Typ des pktsnip registriert haben. Daraufhin verschickt die netapi den pktsnip in einer *IPC-Nachricht*, sequenziell an die zum Typ registrierten Threads. Die *IPC-Nachricht* selbst wird entsprechend der Verarbeitungsrichtung des Pakets, Senden oder Empfangen, typisiert. Der benachrichtigte Thread wertet diesen Typ aus und leitet die Verarbeitung des angehängten pktsnip aus der *IPC-Nachricht* ein. Bei der Verarbeitung eines empfangenen Pakets werden in den Schichten die korrespondierenden im pktsnip angehängten Rohdaten des Paketausschnitts ausgewertet. Bei der Auswertung werden die Kontrollinformationen des Pakets gelesen und typisiert. Durch die Bestimmung des Typs im ausgewerteten Paketkopf der Rohdaten des pktsnips, kann die Verarbeitungsfunktion der Schicht jeweils einen neuen pktsnip erzeugen mit dem das Paket weiter unterteilt werden kann. Der neue pktsnip wird an den von der netapi empfangenen pktsnip als Nachfolgender angehängt. Den erzeugten pktsnip wird dann an die

nächsthöhere Schicht mittels `netapi` weitergeleitet, bis der Inhalt des Pakets vollständig den Netzwerkstack durchlaufen hat und verarbeitet wurde.

6.1.2 Verarbeitung beim Versenden eines Pakets

Zum Versenden eines Pakets wird zunächst ein Grundgerüst aus `pktsnips` erzeugt, in dem anwendungsspezifische Informationen, wie zu übertragene Daten, die Zieladresse und der Zielport eingetragen werden. Hierfür bereitet eine Anwendung die ein Paket verschicken will zuerst die zu transportierenden Daten als `pktsnip` des Typs GNRC_NETTYPE_UNDEF vor. Daraufhin wird ein `pktsnip` der Transportschicht erzeugt, der dem zuvor erstellten `pktsnip` für die Daten vorangestellt wird. Dabei wird beispielsweise ein Paketkopf eines UDP Pakets in einem `pktsnip` erzeugt und dem `pktsnip` der Daten vorangestellt. Ein UDP Paketkopf wird beim Erzeugen mit Quell- und Zielport mithilfe der Hilfsfunktion `gnrc_udp_hdr_build()` parametriert. Der `pktsnip`, mit dem hinzugefügten Paketkopf aus der Transportschicht wird seinerseits in einen IPv6 `pktsnip` der Vermittlungsschicht eingebettet. Beim Erzeugen des IPv6 `pktsnips` wird die Quell- sowie Zieladresse mithilfe der Hilfsfunktion `gnrc_ipv6_hdr_build()` parametriert und in den generierten Header der Vermittlungsschicht eingetragen. Der zuletzt erzeugte `pktsnip` der Vermittlungsschicht, in dem der Header der Transportschicht und die Daten als Kette von `pktsnips` eingebettet sind, wird mittels `netapi` an den zuständigen Thread der Transportschicht weitergeleitet. Von der Transportschicht aus vervollständigt jede Schicht nacheinander seinen zugehörigen Header im korrespondierenden `pktsnip`. Mittels `netapi` leitet jede Schicht den Nachfolger ihres verarbeiteten `pktsnip` an den nächsten interessierten Thread weiter, bis das Paket zum Versenden an das `gnrc_netdev2` Modul weitergereicht und abschließend verschickt wird.

6.2 Eingehende RPL Kontrollnachrichten

Während der Initialisierung registriert sich RPL bei der `netreg` von RIOT bei eingehenden ICMPv6 Kontrollnachrichten des Typs 155 benachrichtigt zu werden. RPL startet daraufhin einen Thread, der auf eingehende *IPC-Nachrichten* des registrierten Typs wartet und diese anhand ihres Typs weiterverarbeitet. Beim Empfang eines durch den Typ 155 gekennzeichneten RPL spezifischen ICMPv6 Pakets, benachrichtigt die `netapi` immer den RPL Thread mittels einer *IPC-Nachricht*. Dabei beinhaltet die *IPC-Nachricht* einen `pktsnip`, der auf den ICMPv6 Header des empfangenen Paktes zeigt. Beim Empfang einer *IPC-Nachricht* liest der RPL Thread die Informationen aus dem Header der ICMPv6 Kontrollnachricht aus und verteilt sie anhand des Eintrags im Code-Feld an konkrete Verarbeitungsfunktionen seiner Implementierung. Jede der aufgerufenen Funktionen überprüft die angehängten Parameter der Kontrollnachricht und ändert anhand der Informationen Einstellungen, die eine Zustandsänderung des Knotens einleiten. Abschließend werden Änderungen des Zustands auf dem RPL Knoten an Nachbarknoten annonciert.

Eingehende DIO Zuerst wird die Länge der DIO-Kontrollnachricht validiert und die Verarbeitung abgebrochen, falls die erwartete Länge der Nachricht ihre tatsächliche Länge übersteigt. Dann werden die annoncierten Optionen der DIO hinsichtlich des erwarteten Typs überprüft. Dabei erwartet ein RPL Knoten als Optionen in der DIO eine DODAG Konfiguration, eine Präfix-Information oder leere Bytes zum Ausrichten der Kontrollnachricht auf die Grenzen der Speicherbereiche eines Pakets. Nach erfolgreicher Überprüfung legt der Knoten einen neuen Eintrag für die annoncierte RPL Instanz aus der DIO an und trägt den annoncierten MOP sowie die einzusetzende OF für die Instanz ein. Anschließend initialisiert der Knoten den DODAG der neuen Instanz, übernimmt die annoncierte Konfiguration der DIO für den DODAG und startet einen zugehörigen *trickle-timer*. Hat der Knoten bereits einen Eintrag für die Instanz und dessen DODAG, überspringt er das Anlegen sowie dessen Konfiguration. Die annoncierte DODAG Versionsnummer aus der DIO, wird gegen die DODAG Versionsnummer im gespeicherten Tupel für die aktuelle DODAG-Version des RPL Knotens verglichen und übernommen, falls sie aktueller ist. Befindet sich der Knoten bereits in einer aktuelleren DODAG Version, startet er eine lokale Reparatur, um die inkonsistente DODAG Version des Versenders der DIO aufzulösen. Der eingetragene Rang der DIO-Kontrollnachricht, wird dann gegen den eigenen Rang verglichen. Der Versender der DIO-Kontrollnachricht, wird in die Liste der Eltern für den DODAG eingetragen, wenn sein annoncierter Rang kleiner als der aktuelle Rang des Knotens ist und seine geführte Liste der verfügbaren Elternknoten noch Platz für einen weiteren Eintrag hat. Wurde ein neuer Elternknoten in die Liste eingefügt, sortiert sie der RPL Knoten aufsteigend nach dem Rang jedes Eintrags. Der oberste Listeneintrag stellt den bevorzugten Elternknoten des Knotens für den DODAG dar. Daraufhin wird der *default next-hop* zum Root des DODAGs in der *Forwarding Information Base* (FIB) auf die link-lokale Adresse des bevorzugten Elternknotens und seine Lebensdauer aktualisiert. Ändert sich der Rang des bevorzugten Elternknotens nach der Umsortierung oder durch den annoncierten Rang der DIO, bestimmt der Knoten mithilfe der OF seinen eigenen Rang neu und setzt den *trickle-timer* für den DODAG zurück. Das Zurücksetzen des *trickle-timers* bewirkt, dass der Knoten nach dem Verarbeiten der eingegangenen DIO seinerseits DIOs versendet in die er Informationen zu seinem neuen Rang und die angepassten DODAG Konfigurationen einträgt. Wird der DODAG im Storing MOP betrieben, annonciert der Knoten seine *downward* erreichbaren Präfixe an den neuen bevorzugten Elternknoten. Empfängt der Knoten eine DIO von einem seiner eingetragenen Elternkonten aus der Liste der Eltern, überprüft er seine eingetragene DTSN. Wurde die DTSN von dem Elternknoten erhöht, aktualisiert der Knoten die DTSN für den Elternknoten und initiiert ein verzögertes Versenden einer DAO an ihn. Hierfür bereitet er eine DAO vor in der er jedes von ihm aus erreichbare Präfix in *downward* Richtung als `Target Options` mit Transit Information anhängt und an ihn versendet.

Auswirkungen auf den Knoten:

- Neue Instanz wird angelegt
- DODAG Informationen werden gespeichert
- Aktualisierung der gespeicherten Elternknoten wird durchgeführt

- Aktualisierung der Einträge in der FIB
- Neubestimmung des eigenen Rangs
- Zurücksetzten des *trickle-timer*s
- Versenden von DIO-Kontrollnachrichten mit den angepassten Einstellungen
- Versenden von DIS-Kontrollnachrichten im Fehlerfall
- Versenden von DAO-Kontrollnachrichten beim Wechsel des Elternknotens

Eingehende DIS Zu Beginn wird die Länge der DIS-Kontrollnachricht validiert und die Verarbeitung abgebrochen, falls die erwartete Länge der Nachricht ihre tatsächliche Länge übersteigt. Die RPL Implementierung von RIOT wurde dahin gehend vervollständigt, dass DIS-Kontrollnachrichten eine `Solicited Information` Option angehängt und von RPL Knoten verarbeitet werden kann.
Wurde die DIS per Unicast an den Knoten verschickt, überprüft der Knoten ob eine `Solicited Information` angehängt wurde und liest die Filter-Kontrollbits für die DODAG Versionsnummer (V), die `RPLInstanceID` (I) und die DODAG ID (D) der Option aus. Anhand der Kontrollbits wird vom Knoten ermittelt ob er auf die DIS antwortet, indem er die geforderten Übereinstimmungskriterien überprüft. Stimmen die Kriterien überein, oder hatte die DIS keine angehängte `Solicited Information`, antwortet er mit einer DIO per Unicast. In die DIO trägt er dabei immer seine aktuelle DODAG Konfiguration ein.
Wurde die DIS hingegen auf der Multicast-Adresse empfangen, wird der *trickle-timer* des Knotens für den DODAG der Instanz zurückgesetzt und damit ein lokaler Reparaturvorgang eingeleitet. Dabei werden alle eingetragenen Eltern im DODAG des Knotens sowie die zugehörigen Einträge der FIB verworfen. Der Knoten wartet dann auf DIOs von seinen Nachbarn, um neue Elternknoten einzutragen.

Auswirkungen:

- Zurücksetzen des *trickle-timer*s
- Versenden von DIO-kontrollnachrichten

Eingehende DAO Zuerst wird die Länge der DAO-Kontrollnachricht validiert und die Verarbeitung abgebrochen, falls die erwartete Länge der Nachricht ihre tatsächliche Länge übersteigt. Daraufhin vergleicht der Knoten die `RPLInstanceID` aus der DAO mit seinen Instanzen. Ist der Knoten in der gleichen Instanz, zu der die DAO versendet wurde, verarbeitet er die Kontrollbits und Optionen der DAO weiter. Zuerst überprüft er, ob das D-Kontrollbit in der DAO gesetzt ist, und bricht die Weiterverarbeitung der Kontrollnachricht ab, wenn es der Fall ist. Nach der Überprüfung werden die Optionen der DAO verarbeitet, falls der Knoten nicht als Blattknoten im DODAG teilnimmt. Als Einträge erwartet RPL eine oder mehrere `Target`

Options gefolgt von zugehörigen Transit Informations. Für einen Zieleintrag einer Target Option, wird ein Eintrag in der FIB erstellt, in dem zunächst die Adresse des Versenders der DAO als *next-hop* zum Zielpräfix der Option eingetragen ist. Die darauf folgende Transit Information, stellt den tatsächlichen *next-hop* für das Zielpräfix des FIB Eintrags ein. Daraufhin antwortet der Knoten mit einer DAO-ACK-Kontrollnachricht, wenn das K-Kontrollbit der empfangenen DAO gesetzt ist und damit den Knoten auffordert, die Anfrage optional zu bestätigen.

Auswirkungen:

- Aktualisierung der Einträge in der FIB
- Versenden von DAO-Kontrollnachrichten
- Versenden von DAO-ACK-Kontrollnachrichten

Eingehende DAO-ACK Zu beginn wird die Länge der DAO-ACK-Kontrollnachricht validiert und die Verarbeitung abgebrochen, falls die erwartete Länge der Nachricht ihre tatsächliche Länge übersteigt. Daraufhin prüft der Knoten, ob die eingetragene RPLInstanceID der DAO-ACK passend zur eigenen Instanz ist. Ist das D-Kontrollbit der DAO-ACK gesetzt, gleicht der Knoten die DODAGID mit dem DODAG der Instanz ab. Anschließend wird der eingetragene Status sowie die Sequenznummer vom Knoten überprüft. Nach erfolgreicher Überprüfung speichert er den Empfang der DAO-ACK und initiiert ein verzögertes Versenden von DAOs. Wird eine DAO-ACK auf der Multicast-Adresse empfangen verwirft der Knoten die Kontrollnachricht.

Auswirkungen:

- Versenden von DAOs

6.3 Analyse der Verarbeitungskette der Kontrollbits im Options-Header von Datenpaketen

Die IPv6 Implementierung im RIOT Netzwerkstack berücksichtigt derzeit keine Hop-by-Hop Optionen in empfangenen Paketen, die der Knoten als Router in Richtung der eingetragenen Zieladresse des Pakets weiterleitet. Um die Verarbeitung der Kontrollbits zu ermöglichen, wurde die bisherige Weiterleitung von Paketen im RIOT Netzwerkstack angepasst, sodass Hop-by-Hop Optionen verarbeitet werden und RPL über eintreffende und ausgehende IPv6 Datenpakete von der netapi benachrichtigt wird. Der RPL Thread registriert zusätzlich zum Interesse an ICMPv6 Kontrollnachrichten des Typs 155, ein Interesse an IPv6 Paketen, die eine Hopy-by-Hop Header Option beinhalten bei der netreg. Diese werden nach dem Empfang und der Verarbeitung durch das gnrc_netdev2 Modul als pktsnip an den gnrc_IPv6 Thread weitergereicht. Der gnrc_IPv6 Thread verarbeitet daraufhin die Informationen aus

dem Header des IPv6 Pakets. Ist ein Hop-by-Hop Optionsheader im Paket eingetragen, leitet der gnrc_IPv6 Thread den pktsnip des Hop-by-Hop Headers mittels netapi an RPL sowie alle weiteren an diesem Typ interessierten Threads weiter. Der RPL Thread verarbeitet die Nachricht und überprüft, ob eine Hop-by-Hop Option im IPv6 Header des Pakets zu finden ist. Wird die Option gefunden, reicht RPL den pktsnip des Hop-by-Hop Headers an die Funktion gnrc_rpl_hop_opt_process() des neuen gnrc_rpl_hop Moduls zur Auswertung weiter. In der Funktion wird zuerst das F-Kontrollbit im Header überprüft und, wenn es gesetzt ist, die Weiterverarbeitung des Headers abgebrochen indem die Funktion mit dem Rückgabewert HOP_OPT_ERR_INCONSISTENCY zurückkehrt.
Setzt die Funktion die Auswertung fort, wird die eingetragene InstanceID gegen die Instanzen abgeglichen, in denen der Knoten teilnimmt. Stimmt die gegebene InstanceID mit einer der Instanzen des Knotens überein, werden die übrigen eingetragenen Informationen des Headers der Reihe nach ausgewertet. Zuerst wird das O-Kontrollbit ausgewertet und daraus die erwartete Richtung, in der das Paket an den Knoten weitergeleitet wurde ermittelt. Damit gleicht der Knoten seinen im DODAG der Instanz angenommenen Rang, gegen den eingetragenen Rang des Headers und der erwarteten Richtung ab. Wurde das Paket in entgegengesetzter Richtung weitergeleitet, überprüft der Knoten, ob das Paket zuvor schon einmal entgegengesetzt zur erwarteten Richtung propagiert ist, indem das R-Kontrollbit ausgewertet wird. Ist dies der Fall, setzt die Funktion das F-Kontrollbit des Headers, startet eine lokale Reparatur für den DODAG der Instanz und kehrt mit HOP_OPT_ERR_FLAG_F_SET als Rückgabewert zurück. Andernfalls wird das R-Kontrollbit des Headers gesetzt und die Funktion kehrt mit HOP_OPT_ERR_FLAG_R_SET als Rückgabewert zurück. Ist die Überprüfung des Hop-by-Hop Headers abgeschlossen, ohne dass Kontrollbits verändert werden, kehrt die Funktion mit HOP_OPT_SUCCESS zurück. Damit wird RPL angezeigt, dass der pktsnip durch den Netzwerkstack weitergeleitet werden kann. Konnte der Header keiner Instanz des RPL Knotens zugeordnet werden, kehrt die Funktion mit dem Rückgabewert von HOP_OPT_ERR_NOT_FOR_ME zurück.
Kehrt die Auswertung der Kontrollbits mit HOP_OPT_ERR_FLAG_F_SET zurück, speichert der Knoten einen Zähler für den ursprünglichen Versender des Pakets oder erhöht ihn falls er schon vorhanden ist. Bei jeder festgestellten Inkonsistenz leitet der Knoten einen lokalen Reparaturvorgang ein. Empfängt der Knoten ein Paket, in dem das F-Kontrollbit bereits gesetzt ist, leitet er es in Richtung des ursprünglichen Versenders weiter.
Ist hinter dem ausgewerteten Header kein weiterer Header im IPv6 Paket, kann RPL das Paket an den next-hop Richtung Zieladresse weiterleiten. Sind weitere Header hinter dem ausgewerteten Header, leitet RPL die Weiterverarbeitung der Header ein indem eine *IPC-Nachricht* an die Vermittlungsschicht versendet wird. Die *IPC-Nachricht* enthält den pktsnip der Vermittlungsschicht, den pktsnip zum nächsten Header, die Interface Thread PID auf dem das Paket empfangen wurde und die Protokollnummer des nächsten Headers. Damit kann der IPv6 Thread der Vermittlungsschicht die Auswertung der nachfolgenden Header fortsetzen. Hierfür wird per netapi der nächste Header der *IPC-Nachricht* zur Auswertung an alle interessierten Threads versendet. Diese müssen genau wie der RPL Thread nach der Auswertung eine *IPC-Nachricht* an die Vermittlungsschicht senden, in der sie den Zeiger vom pktsnip des nachfolgenden Headers eintragen und mittels netapi an den IPv6 Thread

der Vermittlungsschicht weiterreichen. Ist kein weiterer `pktsnip` zum nächsten Header in der *IPC-Nachricht* eingetragen, versendet der IPv6 Thread das Paket weiter zum next-hop in Richtung der Zieladresse.

Auswirkungen:

- Löschen von FIB Einträgen
- Löschen von eingetragenen DAO Routen
- Versenden von DAO-Kontrollnachrichten
- Einleiten eines lokalen Reparaturvorgangs

7 Laufzeitüberwachung zum Absichern des Routings in RIOT

In der Literaturrecherche aus Abschnitt 4.5 wurden Schwachstellen von RPL vorgestellt, und in den Arbeiten der Autoren ausgenutzt, um das Routing in RPL und den DODAG zu stören, indem sie Knoten erfolgreich mit manipulierten Paketen täuschten. Als Schutz vor den Angriffen oder um die Wirkung eines Angriffs abzuschwächen, schlagen die Autoren in einigen der Arbeiten teils spezialisierte Lösungen und Implementierungen vor. Die anschließende Analyse der ausgenutzten Schwachstellen von RPL in Kapitel 5 zeigte die konkreten Bedingungen mit denen die Gefährdungsszenarien durch einen Angreifer evaluiert werden können. Dabei wurde deutlich, dass Angriffe auf RPL aufgrund von Schwächen in der Legitimation und Verifizierbarkeit von empfangenen Informationen aus Paketen möglich sind. Neu gelernte und potenziell manipulierte Informationen können als Reaktion, Zustandstransitionen auf dem Knoten verursachen und damit direkt Einfluss auf sein Verhalten ausüben. Die dabei wirkenden einzelnen Zustandsveränderungen des Knotens, wie sie anhand der konkreten RPL Implementierung von RIOT in Kapitel 6 vorgestellt wurden, geben Anhaltspunkte ein Fehlverhalten von Nachbarknoten oder sogar potenzielle Angriffe zu erkennen. Zunächst müssen die eingeleiteten Transitionen auf einem Knoten überwacht und ausgewertet werden können, sodass er lokal die dafür verantwortlichen empfangenen Informationen in Beziehung setzen kann. Aus den Erkenntnissen der überwachten Transitionen kann der Knoten Rückschlüsse und Indikatoren ableiten, die ihm potenziell eine Bewertung erlauben ob die Zustandstransition legitim, ein Angriff, oder ein Fehlverhalten eines Nachbarknotens ist.

7.1 Diskussion möglicher Strategien einer Laufzeitüberwachung

Grundsätzlich muss die Laufzeitüberwachung alle von RPL verarbeiteten Informationen lesen können um deren Inhalt auszuwerten. Gleichzeitig muss sie auf den aktuellen Zustand von RPL zugreifen können um festzustellen, ob die empfangenen Informationen Einfluss auf den Zustand nehmen. Dabei muss sie alle in RPL eingesetzten Komponenten und Variablen während der Laufzeit überwachen können. Ein weiteres Kriterium ist, dass zusätzlich zur Kontrolle und Erfassung von Zustandsverändernden Informationen, eine aktive Reaktion seitens der Laufzeitüberwachung durchgeführt werden kann, bevor die tatsächliche Zustandsänderung auf dem RPL Knoten eintritt. Um das zu erreichen kann die vorhandene Implementierung der einzelnen Komponenten von RPL so erweitert werden, dass die Laufzeitüberwachung beim Ausführen jeder einzelnen Komponente aufgerufen wird. Dabei kann entweder die Laufzeit-

überwachung aus der Komponente heraus aufgerufen werden, oder sie substituiert und kapselt die zu überwachenden Komponenten in RPL. In beiden Fällen wird die Verarbeitung der empfangenen Informationen immer zuerst an die Laufzeitüberwachung zur Analyse weitergereicht, bevor diese sie zurück an die Komponente leitet um tatsächlich eine Zustandsänderung auf dem Knoten auszulösen.
Ein Aufruf zur Zustandsänderung innerhalb einer RPL Komponente, bspw. das Zurücksetzen des *trickle-timer*s, kann durch unterschiedliche Ereignisse ausgelöst werden, die von internen Abläufen oder durch die Auswertung von Informationen eintreffender Pakete für RPL initiiert werden. Die Integration von Schnittstellen in die Implementierung einer Komponente von RPL, muss daher den aktuellen Zustand des Knotens ermitteln können, um einerseits die Zustandsänderungen vollständig erfassen zu können und andererseits konkrete Sicherungsmaßnahmen zum Schutz aufgrund der erkannten Änderungen aufzurufen. Dadurch muss die eingesetzte Schnittstelle in der Komponente die Zustandsänderung und den aktuellen Zustand zur Auswertung an die Laufzeitüberwachung weiterleitet. Analog zur Anzahl von integrierten Schnittstellen in Komponenten der RPL Implementierung, würde damit die Zahl der möglichen Permutationen von Zuständen ansteigen, die von der aufgerufenen Laufzeitüberwachung differenziert werden müssen.
Das Ersetzen der angreifbaren Komponenten von RPL durch jeweils eine substituierende Komponente der Laufzeitüberwachung, müsste die ursprüngliche Implementierung der RPL Komponente einbetten. Dabei müsste die Granularität für die gekapselte Komponenten so gewählt werden, dass die Komplexität der Implementierung von RPL und der Mehraufwand zur Kommunikation zwischen den Komponenten vorzugsweise minimal ausfällt. Eine substituierte Komponente würde dabei die Funktionalität der gekapselten Komponente von RPL durch eine eigene Implementierung ersetzen, in der Auswertungen und Gegenmaßnahmen durch die Laufzeitüberwachung implementiert und durchgeführt werden können. Dies resultiert in einer starken Kopplung zwischen der gekapselten Komponente und der Laufzeitüberwachung. Daraus entsteht die Problematik, dass unterschiedliche Ansätze für die Auswertung von Zustandsänderungen und daraus die Einleitung von Gegenmaßnahmen potenziell in der gleichen Komponente implementiert werden müssen oder über verschiedene Komponenten hinweg Informationen miteinander konsolidieren müssen. Dies führt potenziell zu einem hohen Integrations- und Wartungsaufwand sobald verschiedene Ansätze zum Schutz eingesetzt werden sollen.
Die beiden genannten Vorgehen zur Integration der Laufzeitüberwachung in RPL zeigen Schwächen durch ihre enge Bindung an die Implementierung von RPL. Daher muss die Laufzeitüberwachung, als eigenständige Einheit zum Erfassen von zustandsverändernden Ereignissen unabhängig von der RPL Implementierung und seiner Komponenten implementiert sein.
Die hier entwickelte Laufzeitüberwachung soll verhindern, dass eingehende Pakete direkt Einfluss auf den Zustand des RPL Knotens nehmen können, bevor die Auswirkungen überprüft und potenzielle Maßnahmen zum Schutz angewendet werden. Dafür müssen die Zustandstransitionen eines RPL Knotens, losgelöst von der Verarbeitung eintreffender Pakete für RPL stattfinden. Hierfür soll die Laufzeitüberwachung zwischen der Verarbeitung von Paketen für RPL und den daraus eingeleiteten Zustandstransitionen des Knotens, in die RPL Implementierung von RIOT integriert werden. Dort platziert kann sie Zustandsänderungen

erfassen, aktiv in die Verarbeitung von Paketen durch RPL Eingreifen, um Schutzmaßnahmen durchzuführen und erkannte Angriffe sowie ausgelöste Zustandsänderungen zu protokollieren. Dadurch entsteht eine strikte Unterteilung der Verarbeitungskette von eintreffenden Paketen für RPL und ermöglicht die beinhalteten Informationen sowie daraus erwartete Transitionen des Knotens differenziert auszuwerten. Implementierte Maßnahmen zum Schutz können so aus der Laufzeitüberwachung heraus initiiert werden, ohne direkt in die Implementierung der Verarbeitungsfunktionen von RPL eingreifen zu müssen. Es wird dadurch möglich implementierte Maßnahmen, ab hier Sicherungsmodule genannt, modular entwickeln und spezifisch auf Anwendungsszenarien zugeschnitten einsetzen zu können.

7.2 Entwurf der Laufzeitüberwachung

Die grundlegende Idee der Laufzeitüberwachung ist ein strukturiertes Vorgehen beim Erkennen von Angriffen und daraus dem Einleiten von Gegenmaßnahmen zu organisieren. Die Laufzeitüberwachung für RPL wird an den Anfang der Verarbeitungskette von eintreffenden `pktsnips` im RPL Thread platziert. Sie substituiert die Ereignisverarbeitung des RPL Threads in der eingetroffene `pktsnips` in *IPC-Nachrichten* von der netapi, hinsichtlich ihres Inhalts klassifiziert, und zur Weiterverarbeitung an spezialisierte Funktionen verteilt werden. Durch die Substitution der Ereignisverarbeitung werden alle eingehenden `pktsnips` für die sich der RPL Thread bei der `netreg` registriert hat, an die Laufzeitüberwachung übergeben. Dort werden sie durch die Laufzeitüberwachung verarbeitet. Damit durchlaufen alle für RPL relevanten `pktsnips` aus empfangenen Paketen zuerst die Laufzeitüberwachung, bevor diese sie zur endgültigen Verarbeitung an konkreten Funktionen von RPL weiterreicht, vgl. Abb. 7.1.
Die grundlegende Struktur der Laufzeitüberwachung ist in folgende Abläufe unterteilt:

1. Erkennung von zustandsändernden Informationen eintreffender Pakete auf dem RPL Knoten
2. Aufbereiten der erkannten Änderungen für die Weiterverarbeitung
3. Auslösen von dedizierten Schutzmaßnahmen für die Zustandsänderungen
4. Protokollieren der Ereignisse
5. Auslösen der Zustandstransition des RPL Knotens

Zur Realisierung des aufgeführten Ablaufschemas ist die Laufzeitüberwachung in zwei logische Einheiten unterteilt, die miteinander interagieren. Die beiden logischen Einheiten unterteilen die Laufzeitüberwachung in eine Identifikations- und eine Reaktionseinheit.
Die Identifikationseinheit dient zum Erfassen von zustandsverändernden Informationen eintreffender Pakete auf den RPL Knoten. Anders als die vorgestellten spezialisierten Lösungen zum Absichern von RPL aus Kapitel 4.5, wird durch die Laufzeitüberwachung eine Abstraktionsebene in RPL eingeführt, die losgelöst von einem konkreten Gefährdungsszenario, Auswirkungen auf den RPL Knoten feststellt. Dabei überwacht die Laufzeitüberwachung bei eingehenden

Paketen die potenziellen Auswirkungen auf die in Kapitel 6 identifizierten internen Zustände des RPL Knotens. Festgestellte Auswirkungen auf den RPL Knoten werden von ihr erfasst und zur Weiterverarbeitung bereitgestellt.
Die Reaktionseinheit dient zum Ausführen von Aktionen bedingt durch die erfassten Auswirkungen der Identifikationseinheit. Sie ruft Sicherungsmodule auf, in denen konkrete Implementierungen von Gegenmaßnahmen implementiert sind. In der Reaktionseinheit können Implementierungen von Gegenmaßnahmen, bspw. die in Kapitel 4.5 vorgestellten Ansätze, spezifisch erkannten Auswirkungen zugeordnet werden. Die Reaktionseinheit bestimmt, ob die festgestellten und überprüften zustandsverändernden Informationen aus empfangenen Paketen für RPL weiterverarbeitet werden. Dadurch legitimiert sie das Einleiten der abschließenden Zustandstransition des RPL Knotens.
Die Laufzeitüberwachung koordiniert dabei die beiden logischen Einheiten.
Nachfolgend werden die einzelnen Komponenten der Laufzeitüberwachung näher diskutiert.

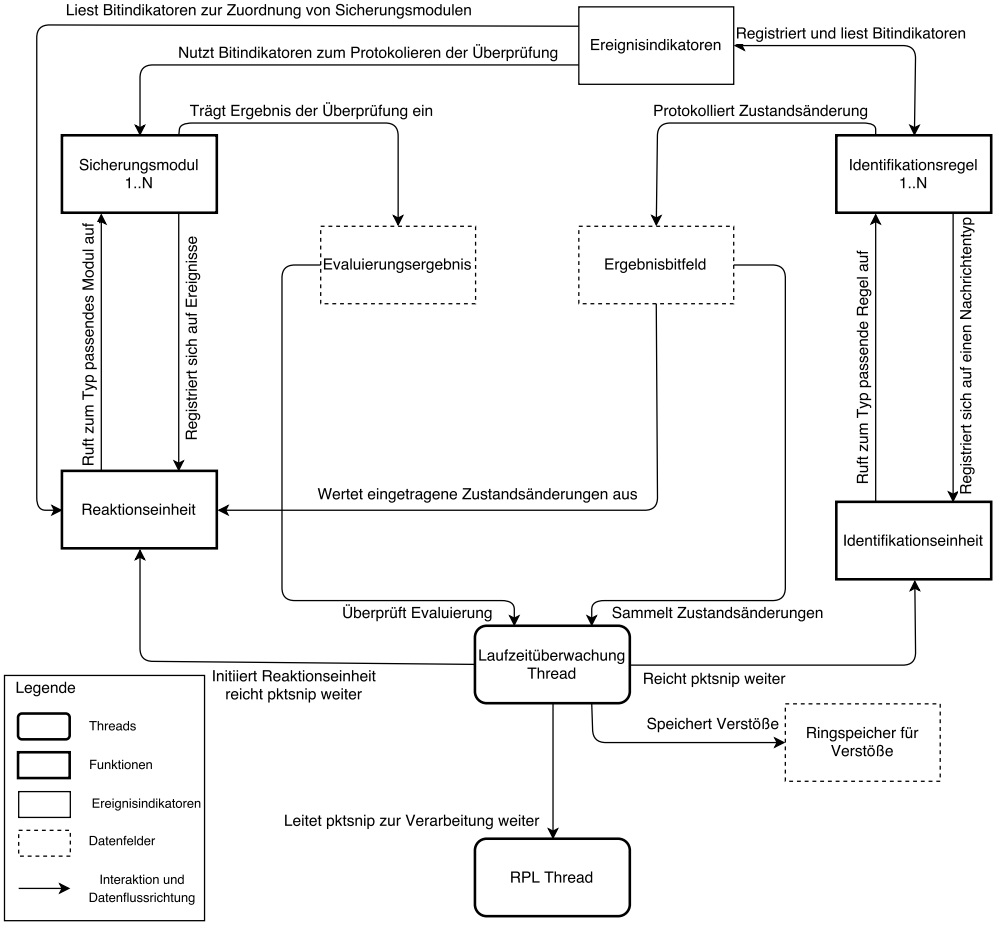

Abbildung 7.1: Übersicht der Laufzeitüberwachung

7.2.1 Identifikationseinheit

Die Identifikationseinheit der Laufzeitüberwachung verarbeitet die Informationen aus allen für RPL eingetroffenen Paketen auf dem Knoten. Dies beinhaltet sowohl die Informationen aus RPL Kontrollnachrichten, als auch aus der *Hop-by-Hop Option* eines IPv6 Paketkopfs von Paketen aus der Datenebene. Beim Auswerten eines Pakets wird generell von der Identifikationseinheit festgestellt, ob die eingetragenen Informationen eine Zustandsänderung auf dem RPL Knoten hervorrufen. Dabei überprüft sie, welche Auswirkung jede Information im empfangenen Paket auf die in Kapitel 6 identifizierten internen Zustände des RPL Knotens hat. Für eine strukturierte Auswertung der Informationen aus einem empfangenen Paket für RPL, wird die Überprüfung der Pakete oder spezifischer Felder in kleinere Einheiten unterteilt. Sie werden von der Identifikationseinheit sequenziell auf das Paket angewendet. Diese Einheiten, ab hier Identifikationsregeln genannt, beinhalten konkrete Implementierungen, mit denen sie die resultierende Auswirkung auf den Zustand des Knotens durch eingegangene Informationen in Paketen für RPL ermitteln. Das Ergebnis der Überprüfung, ob die ausgewertete Information den Zustand des RPL Knotens verändert und welche konkrete Änderung eintreten würde, wird von jeder Identifikationsregel global in der Laufzeitüberwachung zur Weiterverarbeitung und Auswertung erfasst. Der aktuelle Zustand eines RPL Knotens wird durch die konkret eingetragenen Werte seiner Variablen beschrieben. Zum Erfassen einer zustandsverändernden Information aus einem Paket, greift die Identifikationseinheit auf Variablen des Knotens zu und bestimmt die kommende Veränderung der Variablen. Dabei werden alle kommenden, kausal abhängigen Zustandsänderungen der Variablen des Knotens von ihr erfasst. Die global in der Laufzeitüberwachung protokollierten Änderungen beschreiben damit ein Delta zum aktuellen Zustand des RPL Knotens. Die Identifikationseinheit führt keine Bewertung des ursprünglichen, oder des kommenden Zustands durch, sondern erfasst ausschließlich das Delta. Alle erfassten Konstellationen der Variablenänderungen beschreiben letztendlich eine Zustandstransition, die durch das empfangene Paket auf dem RPL Knoten ausgelöst wird. Da die Zustandstransition noch nicht eingeleitet ist, kann die konkrete Änderung nach der Bestimmung des Deltas genauer, mithilfe von zugeschnittenen Maßnahmen für den ermittelten Zustandswechsel, untersucht und protokolliert werden.

7.2.2 Identifikationsregeln

Eine Identifikationsregel wird jeweils durch eine Funktion implementiert, mit der die Informationen aus dem `pktsnip` eines eingegangenen Pakets für RPL überprüft werden. Identifikationsregeln werden bei der Identifikationseinheit registriert, um auf die Informationen des eingegangenen Pakets angewendet zu werden. Zur Zuordnung einer Identifikationsregel zu spezifisch transportierten Informationen eines `pktsnip`, wird die Regel zu einem konkreten für sie relevanten Typ registriert. Beim Registrieren einer Identifikationsregel speichert die Identifikationseinheit eine Struktur aus einem definierten Funktionszeiger zur Identifikationsregel und dem Typ der Information, die von ihr überprüft wird, in einer Liste ab. Der Typ entspricht dabei den Typbezeichnungen aus der `netreg` für RPL Kontrollnachrichten oder weiteren Typen wie dem einer *Hop-by-Hop Option* im Header eines Datenpakets. Die Signatur

des Funktionszeigers ist in der Identifikationseinheit festgelegt und wird zum Einleiten der Zustandserfassung implementiert. Durch die verwendete den registrierten Funktionszeiger wird festgelegt, welcher Informationstyp aus dem empfangenen Paket an sie weitergegeben werden kann. Die Signatur des eingetragenen Funktionszeigers ist auf den Typ der Identifikationsregel spezialisiert, sodass nur zum Typ passende Informationen als Parameter an die Funktion weitergegeben werden können. Wird ein Paket durch die Identifikationseinheit überprüft, wendet sie alle auf den Typ des Pakets registrierten Identifikationsregeln sequenziell auf die typisierten Informationen aus dem Paket an. Hierfür überprüft sie, ob der Pakettyp dem eingetragenen Typ einer Identifikationsregel übereinstimmt, und ruft den zugehörigen Funktionszeiger auf, wenn es der Fall ist. Beim Aufrufen der Identifikationsregel wird die typisierte Information aus dem eingegangenen Paket an die Funktion als Parameter übergeben. Es können mehrere Identifikationsregeln auf den gleichen Pakettyp registriert sein. Dies ermöglicht, bei Bedarf, die Erfassung von Zustandsänderungen auf bestimmte Informationen eines Pakets zu spezialisieren. Erkannte Zustandsänderungen werden von einer Identifikationsregel zerstörungsfrei erfasst. Dadurch bleibt das Ergebnis erkannter Änderungen von Variablen und Zuständen auf dem Knoten erhalten, die bereits zuvor durch angewendete Identifikationsregeln festgestellt wurden. Unter der Annahme, dass die Transitionsregeln eines RPL Knotens explizit in den RFCs spezifiziert sind, können dabei keine divergierenden Ergebnisse bei den erfassten Zustandsänderungen auftreten. Durch die typisierte Registrierung von Identifikationsregeln wird festgelegt, welche Pakete durch die Laufzeitüberwachung überprüft werden. Als grundlegende Basis sind die Funktionssignaturen für RPL Kontrollnachrichtentypen und den Hop-By-Hop Optionen festgelegt. Diese können jedoch bei Bedarf um weitere Typen erweitert werden, bspw. wenn zusätzliche Nachrichtentypen für Sicherungsmaßnahmen eingesetzt werden. Durch die Zuordnung des Typs zu einer Identifikationsregel und der Möglichkeit verschiedene Identifikationsregeln für einen gemeinsamen Typ zu registrieren, können die überprüften Zustandsänderungen flexibel auf Anwendungsszenarien der RPL Topologie abgestimmt werden.

Funktionaler Ablauf einer Identifikationsregel Innerhalb der Funktion wird zuerst überprüft, ob die als Parameter übergebene typisierte Information aus dem empfangenen Paket die von der Identifikationsregel zu prüfenden Information enthält. Sind die eingetragenen Informationen ungültig, bspw. wenn Bereichsgrenzen für die erwarteten Einträge nicht eingehalten wurden, wird die Überprüfung abgebrochen und der Grund dafür global in der Identifikationseinheit protokolliert. Wird die Überprüfung fortgesetzt, wertet die Identifikationsregel alle für sie relevanten Werte der Felder des Pakets aus und stellt sie in Bezug zum aktuellen Zustand des Knotens. Hierfür greift die Identifikationsregel auf alle Veränderlichen Variablen des RPL Knotens zu, in denen sein Zustand erfasst ist.
In der derzeitigen RPL Implementierung von RIOT sind diese Variablen in zentralen Datenstrukturen abgelegt, die allen Funktionen von RPL zugänglich sind, sodass diese sie auslesen und verändern können. Ein RPL Knoten fasst die Informationen über seinen Zustand innerhalb einer RPL Instanz und des zugehörigen DODAGs in einer Liste zusammen. Für jede Instanz hält

der Knoten die Instance ID einen Zeiger zur beschreibenden Datenstruktur seines zugehörigen DODAGs sowie die Vorgaben des MOP und einen Zeiger zu der eingesetzten OF. Dabei handelt es sich um feste unveränderliche Größen für die gesamte Lebensdauer einer RPL Instanz. Daneben führt die Datenstruktur der Instanz nummerische Schrittgrößen für eine Rangebene, die zur Laufzeit eingestellt und verändert werden können. Die Datenstruktur für den DODAG einer Instanz beinhaltet als unveränderliche Größen während der Lebensdauer des DODAGs, die DODAG ID und Einstellungen für Zeitintervalle von trickle Parametern. Als veränderliche Größen beinhaltet die Datenstruktur einen Zeiger zu einer Liste von Einträgen der Eltern des Knotens, seinen aktuell genutzten Rang im DODAG, seine aktuell eingesetzte DTSN, die Versionsnummer des DODAGs sowie einen trickle-timer, mit dem interne Abläufe zeitgesteuert eingeleitet werden. Die Liste von Datenstrukturen für Elternknoten ist veränderlich und beinhaltet weitere Informationen zu den Eltern eines Knotens. Der oberste Eintrag in der Liste identifiziert immer den aktuell bevorzugten Elternknoten. Die übrigen Einträge der Liste sind das Eltern-Set des Knotens. Jeder Eintrag für einen Elternknoten beinhaltet veränderliche Variablen in denen die link lokale Ipv6 Adresse des Elternknotens, sein annoncierter Rang, die annoncierte DTSN und Lebensdauer des Eintrags gespeichert werden. Präfixe, die über einen Elternknoten *downward* erreichbar sind, werden in der FIB eingetragen. Dabei wird die link lokale IPv6 Adresse des entsprechenden Elternknotens als next-hop zum Zielpräfix des FIB-Eintrags verwendet. Mit den typisierten Informationen des Pakets und den gespeicherten Werten in den Variablen des aktuellen Zustands wird ermittelt, ob eine Zustandstransition von dem empfangenen Paket ausgelöst wird. Hierfür implementiert eine Identifikationsregel zu prüfende Bedingungen, die abgeleitet von den definierten Transitionsregeln aus den RFCs für Knoten in RPL bestimmt, ob eine Zustandsänderung hervorgerufen wird. Die Granularität der gleichzeitig überprüften Bedingungen, mit der eine Identifikationsregel Zustandsänderungen untersucht, ist unabhängig von der konkreten RPL Implementierung von RIOT, in der die Verarbeitung von Paketen für RPL durchgeführt wird. Jede ausgewertete Information innerhalb eines verarbeiteten Pakets hat eine direkt zugehörige Zustandsbeschreibende Variable auf dem Knoten. Hat die Identifikationsregel aufgrund der Informationen im Paket ein Delta des Wertes in einer Variable erkannt, kann dies ein Auslöser für Veränderungen weiterer Variablen auf dem Knoten sein. Beispielsweise hat das Anheben der DODAG Versionsnummer Auswirkungen das Eltern-Set, dass seinerseits bestimmend für den Rang des Knotens ist und festlegt, welche *downward* Routen durch einen Elternknoten verwaltet werden sollen. Die kausalen Beziehungen der Variablen sowie ausgeführten Aktionen sind ausschlaggebend für die Erfassung einer kommenden Zustandstransition des Knotens. Diese kausalen Abhängigkeiten können in die folgenden Abfolgen zusammengefasst werden:

- Eine neue Instanz wird vom Knoten eingetragen
  - Der DODAG Eintrag und das Eltern-Set für die Instanz wird initialisiert
  - Der Rang des Knotens wird initial auf `INFINITE_RANK` gesetzt
- Ein neuer bevorzugter Elternknoten wird in das Eltern-Set eingetragen
  - Der *default next-hop* in der FIB wird aktualisiert

  - Der Rang des Knotens wird an den Rang des bevorzugten Elternknotens angepasst
- Das Eltern-Set wird nach dem Rang sortiert
  - Der oberste Eintrag im Eltern-Set wird aktualisiert
- Neuer Elternknoten wird in das Eltern-Set aufgenommen
  - Die DTSN, der Rang und die Lebensdauer werden für den neuen Elternknoten gespeichert
  - Das Eltern-Set wird nach dem Rang sortiert
- DODAG Version wird erhöht
  - Das Eltern-Set wird geleert
  - Ein neuer Elternknoten wird in das Eltern-Set aufgenommen
  - Der *trickle-timer* wird für den DODAG zurückgesetzt
- DTSN für einen Elternknoten wird erhöht
  - DAO wird vorbereitet
  - Im Non-String MOP wird die eigene DTSN angeglichen
- DIS-Kontrollnachricht wird per Unicast empfangen
  - Die eingetragenen Filterkriterien in der DIS werden geprüft
  - Eine DIO wird per Unicast geantwortet
- DIS-Kontrollnachricht wird per multicast empfangen
  - Der *trickle-timer* des DODAGs wird zurückgesetzt
  - Das Eltern-Set wird geleert
- DAO-Kontrollnachricht wird per Unicast empfangen
  - FIB Einträge für *downward* Präfixe werden aktualisiert
  - Optional wird eine DAO-ACK geantwortet

Regeln zum Auswerten einer DIO-Kontrollnachricht

- Das `RPLInstanceID` und das `DODAGID`-Feld sowie das G-Kontrollbit der DIO, ordnen die eingetragenen Informationen der auf ihnen nachfolgenden Felder einer bestimmten Topologie zu. Ist der Knoten nicht Teilnehmer der angegebenen RPL Instanz aus der DIO, die durch die `RPLInstanceID` identifiziert wird, ist das Verhalten im RFC nicht definiert. Der Knoten hat die Option die Kontrollnachricht der ihm unbekannten RPL Instanz zu ignorieren oder Teilnehmer der Instanz zu werden. Nimmt er Teil, legt er intern einen neuen Eintrag für die Instanz an und ordnet ihr, wie oben beschrieben, den mit der `DODAGID` identifizierten DODAG zu. Die dabei ausgelösten Änderungen des Zustands werden von der Identifikationsregel in der Laufzeitüberwachung erfasst.

- Das `Version Number`-Feld gibt die Iteration eines DODAGs an. Der Knoten überprüft, ob die eingetragene Versionsnummer im Tupel seiner DODAG Version veraltet ist, wenn er im DODAG des Versenders teilnimmt. Ist die DODAG Versionsnummer des Knotens und die aus der DIO gleich, entsteht keine Zustandsänderung aufgrund des `Version Number`-Feldes der DIO. Sobald die `Version Number` eines Knotens, abweichend zu der annoncierten `Version Number` einer DIO ist, werden Zustandsänderungen auf dem Knoten ausgelöst. Bei einer älteren annoncierten Iteration, antwortet der Knoten mit einer DIO per Unicast an den Versender um ihm aktuellere Informationen zum DODAG mitzuteilen. Die anderen Felder der DIO werden dann nicht weiter ausgewertet, da sie Informationen zu einer veralteten DODAG Version enthalten. Ist hingegen die annoncierte DODAG `Version Number` größer, beginnt der Knoten in die neue DODAG Version zu migrieren. Dieser Vorgang löst wie oben beschrieben eine Kette von Ereignissen aus, die von der Identifikationsregel protokolliert werden.
- Das Rang-Feld der DIO wird vom Empfänger mit den eigenen Rang verglichen. Mit dem Ergebnis dieses Vergleichs kann der Knoten entscheiden, ob der Versender als Elternknoten in Frage kommt. Bei einem größeren Rang werden die folgenden Felder der DIO nicht weiter ausgewertet, da der Versender eine topologisch höhere Distanz zum Root des DODAG hat, und dem Knoten nicht zur Kommunikation *upward* dienen kann. Ist der annoncierte Rang kleiner oder gleich seinem eigenen, kann er ihn als Elternknoten in sein Eltern-Set aufnehmen. Dabei werden die oben beschriebenen Ereignisse auf dem Knoten ausgelöst, die seinen Zustand ändern. Dies wird in der Laufzeitüberwachung protokolliert.
- Das DTSN-Feld gibt die Aktualität von Pfaden zu *downward* erreichbaren Präfixen über einen bestimmten Elternknoten an. Ist der Versender der DIO ein Elternknoten und die DTSN erhöht, antwortet der Knoten mit einer DAO, in die er alle über diesen Elternknoten verwalteten Präfixe *downward* einträgt. Daraufhin wird die DTSN für den Elternknoten an die annoncierte aus der DIO angeglichen.

Beim Auswerten wird in der Laufzeitüberwachung gespeichert, dass eine DIO empfangen wurde.

Regeln zum Auswerten einer DIS-Kontrollnachricht

- Das `Type` und `Opt Length`-Feld einer DIS sind statisch festgelegt. Die Filterkriterien in den `VID` Kontrollbits werden vom Knoten ausgewertet und mit den darauf folgenden `RPLInstanceID`-Feld, sowie die optional angehängte `DODAGID` und der `Version Number` verglichen. Stimmen diese mit den Einträgen des Knotens überein, antwortet er eine DIO an den Versender der DIS.
- Wurde die DIS per Multicast empfangen, wertet der Knoten die DIS als lokale Inkonsistenz, wodurch er sein Elten-Set löscht und einen lokalen Reparaturvorgang einleitet.

Beim Auswerten wird gespeichert, dass eine DIS empfangen wurde.

Regeln zum Auswerten einer DAO

- Bei einer empfangenen DAO wird zuerst das RPLInstanceID-Feld ausgewertet, um früh entscheiden zu können, ob die restliche Information in der DAO ausgewertet werden soll. Mit einem gesetzten K-Kontrollbit wird der Elternknoten aufgefordert, seinem Kind eine DAO-ACK als Empfangsbestätigung für die DAO zu antworten. Der DAOSequence Eintrag wird dann als Identifikationstoken in die Empfangsbestätigung eingetragen. Ist das D-Kontrollbit gesetzt, wird die optionale DODAGID in die frühe Filterung einbezogen. Sind die Filterkriterien erfüllt, wertet der Knoten die nachfolgenden Target Option mit ihrer zugehörigen Transit Information aus. Für jedes neue Präfix das in den Optionen der DAO eingetragen ist, legt der Knoten einen FIB-Eintrag an, in den er als next-hop *downward* zu dem Präfix die link-lokale Adresse des Versenders einträgt. Dabei wird mit dem E-Kontrollbiteiner Tarnsit Information festgelegt, ob das Präfix ein DODAG internes oder ein externes Präfix ist. Im Path Control-Feld gibt der Kindknoten seinem für *downward* Routen gewählten Elternknoten eine Präferenz, mit der er die annoncierten Präfixe für sein Kind bedienen soll. Der Zähler im Path Sequence-Feld wird vom Knoten gespeichert, und an das Präfix gebunden. Dadurch kann er ermitteln, ob eine Aktualisierung des Präfixes durch den Originalversender stattgefunden hat. Das Path lifetime-Feld gibt die Lebensdauer des FIB Eintrags an. Für jede Transit Information wird der FIB Eintrag erstellt, oder die Lebensdauer eines vorhandenen Eintrags aktualisiert. Dabei kann die Lebensdauer auf 0 zum Löschen des Eintrags oder auf unendlich im Path lifetimeFeld angegeben werden.

Regeln zum Auswerten einer DAO-ACK

- Beim Auswerten einer empfangenen DAO-ACK werden zunächst die RPLInstanceID und falls das D-Kontrollbit gesetzt ist die angehängte DODAGID ausgewertet. Daraufhin wird die DAOSequenceNumber mit den gespeicherten DAOSequenceNumber für eine zuvor versendete DAO-Kontrollnachricht verglichen. Das darauf folgende Status Feld gibt dem Knoten die Rückmeldung seines Elternknotens, ob es *downward* Präfixe, für ihn bedienen kann. Zwischen den Werten 0-127 stimmt der Elternknoten mit abnehmender Präferenz zu, den Präfix zu bedienen. Ab 128-256 verweigert der Elternknoten das Bedienen des *downward* Präfixes mit zunehmender Abneigung bei steigendem Wert.

7.2.3 Ergebnisbitfeld

Das Ergebnisfeld ist eine globale Datenstruktur der Laufzeitüberwachung, in der die Ergebnisse der Überprüfungen von aufgerufenen Identifikationsregeln gespeichert werden. In dem Ergebnisfeld wird das Ergebnis der Auswertung des Pakets sowie sein Typ in einzelnen Bits festgehalten. Jedes Bit eines Ergebnisses ist einem Parameter des RPL Knotens zugeordnet, der durch eingehende Pakete beeinflussbar ist. Bei jedem eintreffenden Paket, das die Laufzeitüberwachung verarbeitet, wird das Ergebnisfeld zurückgesetzt und mit den aktuellen Ergebnissen für das Paket gefüllt.

Ein gesetztes Bit im Ergebnisbitfeld zeigt, dass die Änderung durch die Verarbeitung ausgelöst werden würde. Das Ergebnisbitfeld dient einerseits zum Protokollieren des Empfangs von Zustandsändernden Informationen, und andererseits als Auslöser, konkrete Gegenmaßnahmen der Sicherungsmodule aus der Reaktionseinheit einzuleiten. Die Größe des Ergebnisfelds wird zur Übersetzungszeit festgelegt. Wird eine Identifikationsregel bei der Laufzeitüberwachung registriert, kann sie zusätzliche Ergebnisbits festlegen, in der sie das ermittelte Ergebnis einer überprüften Information abspeichert. Hierfür wird durch eine einkompilierte Identifikationsregel für jedes Ergebnis das nächste freie Bit im Ergebnisfeld reserviert und der Index zum nächsten freien Bit hochgezählt.

7.2.4 Reaktionseinheit

Die Reaktionseinheit der Laufzeitüberwachung nutzt die Ausgewerteten und im Ergebnisbitfeld hinterlegten Zustandsdeltas der Identifikationseinheit, um Sicherungsmodule zum Schutz aufzurufen. Dabei implementiert ein Sicherungsmodul konkrete Maßnahmen, mit denen die empfangenen Informationen aus Paketen für RPL evaluiert und die dadurch verursachten Änderungen des Knotens legitimiert werden können. Jedes Sicherungsmodul registriert sich hierfür für bestimmte Ereignisse. Die Laufzeitüberwachung speichert dabei für jedes Sicherungsmodul eine Datenstruktur, in der ein Funktionszeiger zum Aufrufen des Sicherungsmoduls, und ein Filter zur Zuordnung des Sicherungsmoduls und den von ihm festgelegten Ereignissen eingetragen ist. Der Filter hat die gleiche Wortbreite in Bits wie das Ergebnisbitfeld und legt die signifikanten gesetzten Bits von erfassten Änderungen fest, die relevant für das Sicherungsmodul sind. Immer wenn das Ergebnisbitfeld durch die Identifikationsregeln befüllt wurde, überprüft die Laufzeitüberwachung, ob seine gesetzten Bits mit den relevanten Bits des Filters eines Sicherungsmoduls übereinstimmt. Sobald die Laufzeitüberwachung eine Übereinstimmung feststellt, ruft sie den registrierten Funktionszeiger des Sicherungsmoduls auf und löst die Evaluierung des `pktsnips` durch das Sicherungsmodul aus. Als Ergebnis der Evaluierung des `pktsnips` gibt das aufgerufene Sicherungsmodul ein Bitfeld zurück, dass von gleicher Wortbreite in Bits wie des Ergebnisbitfeld ist. Ein gesetztes Bit in dem zurückgegebenen Bitfeld des Evaluierungsergebnisses gibt an, dass die durch den Bit angezeigte Änderung, einen Verstoß aus Sicht des Sicherungsmoduls darstellt. Nachdem alle Sicherungsmodule die Änderungen des Zustands vom Knoten überprüft haben, überlagert die Laufzeitüberwachung die Bitfelder der einzelnen Evaluierungen zerstörungsfrei zu einem Gesamtergebnis. Gleichzeitig stellt die Reaktionseinheit fest, welche aufgetretenen Änderungen nicht durch Sicherungsmodule abgehandelt werden. Hierfür werden alle Bitfelder der Evaluierungen invertiert und zerstörungsfrei überlagert. Das daraus resultierende Bitfeld wird mit dem Ergebnisbitfeld der Identifikationseinheit zerstörend überlagert. Alle bestehenden Bits nach dieser Überlagerung zeigen die auftretenden Änderungen, für die kein Sicherungsmodul zugeordnet ist. Als Gesamtergebnis entsteht ein Bitfeld in dem nur überprüfte und evaluierte Änderungen festgehalten sind.
Ist in dem Gesamtergebnis ein Bit für eine Änderung gesetzt, wertet die Reaktionseinheit die empfangene Information als Angriffsversuch. Dieser wir protokolliert, indem das Ergebnisbitfeld der Identifikationseinheit, ein Bitfeld mit nicht verarbeiteten Zustandsänderungen sowie das Gesamtergebnis samt der Quelladresse des Pakets zusammen in einem Ringpuffer fester

Länge abgelegt werden. Der Ringpuffer kann jederzeit von Sicherungsmodulen ausgelesen werden und erlaubt es ihnen, Zustandsänderungen des Knotens retrospektiv nachvollziehen zu können. Die Weiterverarbeitung des `pktsnips` durch die RPL Implementierung wird daraufhin anhand des Gesamtergebnisses von der Laufzeitüberwachung geregelt. Ist eines der Bits im Gesamtergebnis gesetzt, kann sie den `pktsnip` verwerfen und damit die Verarbeitung durch RPL und die daraus resultierende Zustandstransition des Knotens verhindern. Dabei ist die Bewertung des Gesamtergebnisses implementierungsspezifisch, und die konkrete Reaktion nicht unveränderbar festgelegt.
Sind hingegen alle Bits im Gesamtergebnis gelöscht, wertet die Laufzeitüberwachung die Zustandsänderung durch das empfangene Paket als legitim. Dann wird die Zustandstransition des RPL Knotens von der Laufzeitüberwachung eingeleitet. Hierfür reicht die den `pktsnip` an die RPL Implementierung weiter.

7.3 Implementierung der Laufzeitüberwachung

Wie im Entwurf konzipiert, ist die Implementierung der Laufzeitüberwachung in Komponenten unterteilt, die einerseits zur Erfassung von kommenden Zustandsdeltas dienen und andererseits präventive Gegenmaßnahmen oder Reaktionen aufgrund des festgestellten Deltas vom aktuellen Zustand des Knotens durchführen. Als Basis für die Implementierung in RIOT wurde der Release-Stand vom Juli eingesetzt2017[1]. Die konkrete Implementierung der Laufzeitüberwachung ist als Modul in RIOT integriert, dem `rpl_watchdog` Modul. Das Modul hat als einzige Abhängigkeit das `gnrc_rpl` Modul von RIOT und bindet es automatisch ein wenn es eingesetzt wird. Damit kann es immer nur in Tandem mit dem RPL Modul eingesetzt werden. Diese Abhängigkeit besteht nur in einer Richtung, sodass das die RPL Implementierung keine Abhängigkeit zum Modul der Laufzeitüberwachung hat. Um die Laufzeitüberwachung einzusetzen, muss das `rpl_watchdog` Modul im Makefile des Projekts eingetragen werden. Wird die Laufzeitüberwachung in RIOT verwendet, greift sie in den Initialisierungsprozess von RPL ein, und verhindert die Registrierung des RPL Threads bei der `netreg` auf eintreffende Nachrichten. Stattdessen registriert sie hierfür einen eigenen Thread, der die sonst durch RPL verarbeiteten Nachrichten empfängt und sie selbst verarbeitet. Die PID des RPL Threads wird dabei an die Laufzeitüberwachung weitergegeben, sodass sie *IPC-Nachrichten* an den RPL Thread versenden kann. Dieser Thread dient als zentrale Komponente der Laufzeitüberwachung. Die dort auflaufenden *IPC-Nachrichten* von der `netapi` werden, wie zuvor in RPL, zuerst hinsichtlich ihres Inhalts eingeordnet. Nach der Zuordnung des `pktsnips` wird er an die Implementierung der Identifikationseinheit weitergegeben. In der Identifikationseinheit sind Verwaltungsfunktionen zum Registrieren, Abmelden und Aufrufen von konkreten Identifikationsregeln implementiert. Hierfür definiert die Identifikationseinheit eine spezifische Struktur zu einer verfügbaren Funktion und einer Variablen, die durch eine Identifikationsregel implementiert sein muss, vgl. Listing 7.1.

[1] https://github.com/RIOT-OS/RIOT/tree/2017.07-branch

Listing 7.1: Datenstruktur zum Registrieren einer Identifikationsregel

```
typedef struct {
  uint16_t code;
  union {
    int (*apply_dis)(gnrc_rpl_dis_t *pkt);
    int (*apply_dio)(gnrc_rpl_dio_t *pkt);
    int (*apply_dao)(gnrc_rpl_dao_t *pkt);
    int (*apply_dao_ack)(gnrc_rpl_dao_ack_t *pkt);
  } func;
}stRule;
```

Diese Struktur definiert als erstes eine `code` Variable, die den Typ einer Nachricht spezifiziert, für die eine Identifikationsregel ausgelegt ist. Auf diese folgt ein Funktionszeiger, der durch die Identifikationsregel implementiert wird und einen bestimmten Typ von Kontrollnachricht als Parameter übergeben bekommen kann. Beides wird in der beschriebenen Datenstruktur durch eine Identifikationsregel implementiert und mittels angebotener Verwaltungsfunktionen in der Identifikationseinheit registriert. Ein von der Laufzeitüberwachung empfangener und an die Identifikationseinheit weitergegebener `pktsnip` wird zunächst hinsichtlich der erwarteten Größe validiert, die zum ausgelesenen Nachrichtentyp passen muss. Nach dieser Eingangskontrolle übergibt die Identifikationseinheit den typisierten `pktsnip` zur Analyse an alle Identifikationsregeln, die sich zuvor auf den Typ der Nachricht registriert haben. Dafür ruft die Identifikationseinheit den Funktionszeiger jeder registrierten Identifikationsregeln mit dem `pktsnip` als Parameter auf, zu dem der eingetragene Typ passt. Innerhalb einer Identifikationsregel werden die Felder des übergebenen `pktsnips` ausgewertet und mit dem aktuellen Zustand des RPL Knotens verglichen. Hierfür greift eine Identifikationsregel auf die internen Datenstrukturen und Variablen des Knotens lesend zu und stellt mit den Werten fest, ob der die Informationen aus dem `pktsnip` eine Zustandsänderung hervorrufen würden. Wird dabei von der Identifikationsregel eine Änderung in einer der Variablen oder im Zustand des Knotens erkannt, protokolliert die Identifikationsregel das Zustandsdelta in der Laufzeitüberwachung. Zum Sammeln aller erkannten Zustandsdeltas und ihren Auswirkungen auf den RPL Knoten, ist in der Laufzeitüberwachung eine Komponente implementiert mit der die einzelnen Ergebnisse der Auswertungen speicherplatzeffizient gesammelt werden. Sowohl die hervorrufenden Ereignisse als auch die dadurch verursachten Zustandsdeltas werden zentral in der Laufzeitüberwachung jeweils in einem für eine spezifische Änderung in einem Bit codiert und abgespeichert. Die protokollierten Ereignisse und Zustandsdeltas sind in einem Aufzählungstyp als `enum` definiert, deren eingetragener Wert die Bitposition trägt, vgl. Listing 7.2.

Listing 7.2: Zentrale Datenstruktur für die Definition von Ergebnisbits

```
typedef enum {
  eENTRIESBEGIN = 0,
  eDIOpkt,
  ...
  eENTRYCOUNT
}EBitCodes;
```

In der Datenstruktur tragen alle Identifikationsregeln zur Übersetzungszeit ihre verwendeten Bits zum Anzeigen eines erkannten Ereignisses oder einer Veränderung des Zustands ein, bspw. `eDIOpkt` zum Anzeigen des Ereignisses, dass eine DIO-Kontrollnachricht empfangen wurde. Durch die zentral eingetragenen `enum`s sind die festgelegten Bit-Indikatoren für die Identifikationsregeln sowie Sicherungsmodule zugreifbar und erlauben ihnen gleichermaßen die eingetragenen Ereignisse für ihre eigene Registrierung oder Auswertungen zu nutzen. Durch die Nutzung der `enum` Datenstruktur zum Erfassen der verfügbaren Bit-Indikatoren, ist die Anzahl der Einträge automatisch auslesbar, indem der gespeicherte Wert des letzten `enum`s ausgelesen wird (`eENTRYCOUNT`). Hierbei muss der eingetragene Wert der `enum`s automatisch durch den Kompiler festgelegt und der Wert des ersten Eintrags mit 0 festgesetzt werden. Mit der zur Übersetzungszeit festgelegten Anzahl an möglichen Ereignissen und Zustandsdeltas, die in den Bitpositionen der `enum`s codiert sind, wird bei der Initialisierung des Threads der Laufzeitüberwachung automatisch ein Datenfeld passender Größe für die Ergebnisse der Identifikationseinheit angelegt. Zum setzten und testen der Bits im Datenfeld, implementiert die Laufzeitüberwachung Zugriffsfunktionen, mit denen unabhängig von der Breite des Feldes jedes Bit gesetzt, abgefragt und gelöscht werden kann, vgl. Listing 7.3. Für das Setzen eines Bits wird das zu setzende `enum` als Parameter sowie ein Zeiger zum Datenfeld in der das Bit gesetzt werden soll, an eine bereitgestellte Funktion übergeben. Zum Testen ob ein Bit gesetzt ist, wird eine Funktion angeboten an die ebenso das `enum` als Parameter und ein Zeiger zum Datenfeld mit den eingetragenen Bits übergeben wird. Als Rückgabewert gibt diese Funktion an, ob das getestete Bit im Datenfeld gesetzt ist. Als letztes wird eine Funktion zum Löschen eines Bits durch den `enum` Parameter im übergebenen Datenfeld angeboten.

Listing 7.3: Zugriffsfunktionen zum Auslesen und Manipulieren von Bits eines übergebenen Datenfeldes

```
1  void setbit(EBitCodes code, uint8_t* field);
2  bool getbit(EBitCodes code, uint8_t* field);
3  void clearbit(EBitCodes code, uint8_t* field);
```

Somit muss eine Identifikationsregel nicht die Position eines Bits zu einem `enum` kennen, um erkannte Auswirkungen auf den Knoten zu dokumentieren oder eine durch ein spezifisches `enum` eingetragene Änderung des Datenfeldes abzuprüfen. Wenn alle zum Typ der empfangenen Nachricht registrierten Identifikationsregeln aufgerufen wurden, sind in dem Datenfeld alle erkannten Ereignisse und Zustandsdeltas des RPL Knotens, die durch die Verarbeitung des empfangenen Pakets ausgelöst würden, eingetragen. Mit der kondensierten, stark abstrahierten Beschreibung des kommenden Zustandsdeltas, kann die Laufzeitüberwachung differenziert feststellen, ob und welche Sicherungsmodule zum Schutz von RPL aufgerufen werden müssen. Wie die Identifikationsregeln, werden Sicherungsmodule in einer eigenen Komponente, der Reaktionseinheit, registriert und verwaltet. Hierfür definiert die Reaktionseinheit Funktionen, die von einem Sicherungsmodul implementiert und der Reaktionseinheit zur Verfügung gestellt werden müssen, vgl. Lisiting 7.4.

Listing 7.4: Datenstruktur zum Registrieren eines Sicherungsmoduls

```
1 typedef struct {
```

```
2   int (*init)(void);
3   void (*gethandled)(uint8_t* handled);
4   bool (*is_matching)(void);
5   int (*apply)(uint8_t* result);
6 }stProtector_t;
```

Ein Sicherungsmodul muss eine Initialisierungsfunktion bereitstellen, die von der Laufzeitüberwachung beim Starten seines Threads aufgerufen wird, die `init()` Funktion. In dieser Funktion kann ein Sicherungsmodul einleitende Maßnahmen durchführen, bevor der RPL Knoten ein aktiver Teilnehmer einer Instanz wird und beginnt, Kontrollnachrichten mit seinen Nachbarn auszutauschen. Dies kann bspw. von einem Sicherungsmodul genutzt werden, um eine Initialisierung von internen Variablen und Datenfeldern oder das Einleiten eines bootstrapping Vorgangs durchzuführen. Des Weiteren muss von einem Sicherungsmodul eine Funktion implementiert werden, mit der abgefragt wird welche Zustandsänderungen sie Evaluiert, die `gethandled(uint8_t* handled)` Funktion. Sie bekommt als Parameter einen Zeiger auf ein Datenfeld übergeben, dessen Wortbreite in Bits der Anzahl registrierter `enum`s der Identifikationsregeln entspricht. In der Funktion werden von der Identifikationsregel alle enumerierbaren Bits in einem von der Laufzeitüberwachung zentral verwalteten Datenfeld gesetzt, an dem das Sicherungsmodul interessiert ist. Durch ein gesetztes Bit zeigt das Sicherungsmodul an die in dem Bit codierte Änderung zu überprüfen. Dabei werden die Bits zerstörungsfrei in das übergebene Datenfeld eingetragen. Durch das Aufrufen der Funktion für jedes registrierte Sicherungsmodul mit dem Zeiger zu einem gemeinsamen Datenfeld entsteht ein Bitfeld, in dem alle durch die Sicherungsmodule überprüften Zustandsänderungen kodiert sind. Dieses Datenfeld wird zentral im Thread der Laufzeitüberwachung gespeichert. Mit einer weiteren Funktion wird zur Laufzeit das Interesse eines Sicherungsmoduls am erfassten Zustandsdelta durch die Identifikationsregeln abgefragt, der `bool is_matching()` Funktion. Sie greift auf das zentral gespeicherte Ergebnisbitfeld der Identifikationseinheit zu und vergleicht, ob für sie relevante Bits und Bitkombinationen durch Identifikationsregeln gesetzt wurden. Dabei kann jedes Sicherungsmodul die Anzahl an Bits und Bitkombinationen frei definieren, auf die es reagiert. Die Funktion selbst gibt einen boolschen Wert zurück, der für das Sicherungsmodul das Interesse bei true und das Desinteresse an dem erfassten Zustandsdelta bei false angibt. Als letzte erforderliche Funktion muss ein Sicherungsmodul eine Funktion implementieren und anbieten, mit der sie aufgerufen werden kann um ihre konkrete Implementierung von Sicherungsmaßnahmen auszuführen, die `apply(uint8_t* result)` Funktion. Sie bekommt als Parameter, einen Zeiger auf ein gemeinsam von den Sicherungsmodulen genutztes Datenfeld, in das jedes Sicherungsmodul die Ergebnisse der überprüften Änderungen des Knotens einträgt. Das dabei übergebene Datenfeld ist eine Kopie des aktuellen Ergebnisbitfeldes der Identifikationsregeln. In der Funktion wird von dem Sicherungsmodul jedes enummerierbare Bit gelöscht, dessen zugehörige Zustandsänderung überprüft und als unbedenklich einstuft wurde. Nachdem alle Sicherungsmodule das aktuelle Zustandsdelta evaluiert haben, wertet die Laufzeitüberwachung das resultierende Evaluierungsergebnis aus. Abhängig von den eingesetzten Sicherungsmodulen wird nur ein Anteil der erfassten Zustandsdeltas überprüft. Im daraus resultierenden Evaluierungsergebnis können nach der Iteration über alle vorhandenen Sicherungsmodule Bits stehen bleiben, die von keinem Sicherungsmodul über-

prüft wurden. Diese müssen vom Evaluierungsergebnis separiert werden um die tatsächlich erkannten und bedenklichen Zustandsänderungen zu identifizieren. Hierfür wird ein Datenfeld generiert, in dem abschließend alle Zustandsänderungen eingetragen sind, die nicht durch ein Sicherungsmodul überprüft werden. Dabei wird das vorläufige Evaluierungsergebnis der Sicherungsmodule invertiert und mit dem Ergebnisbitfeld der Identifikationseinheit mittels bitweiser Und Operation verglichen. Die einzelnen Ergebnisse des Vergleichs werden in einem neuem Datenfeld gespeichert in dem dann alle nicht überprüften aber aufgetretenen Zustandsänderungen aufgeführt sind. Diese Bits werden aus dem Evaluierungsergebnis der Sicherungsmodule gelöscht. Danach enthält das Evaluierungsergebnis ausschließlich gesetzte Bits für kommende Zustandsänderungen des RPL Knotens, die von den Sicherungsmodulen überprüft und als bedenklich eingestuft wurden. Die insgesamt 3 resultierenden Datenfelder (i) Ergebnisbitfeld, (ii)Evaluierungsergebnis und (iii) die ungeprüften Zustandsänderungen, dokumentieren das kommende Zustandsdelta des Knotens inklusive einer Bewertung des Deltas durch die Sicherungsmodule. Ist in dem Zustandsdelta eine bedenkliche Änderung erkannt worden, werden die 3 Felder von der Laufzeitüberwachung zusammen mit der IPv6 Adresse des Versenders der Kontrollnachricht in einem Ringspeicher der Laufzeitüberwachung festgehalten. In dem aktuellen Eintrag des Ringspeichers wird zusätzlich die Anzahl aller nachfolgenden und als unbedenklich eingestuften Zustandsänderungen eingetragen. Dadurch wird die Häufigkeit von bedenklichen Änderungen erfasst und der Abstand zwischen den bedenklichen Ereignissen in Abständen von verarbeiteten Nachrichten durch den RPL Knoten ablesbar. Der Ringspeicher ist allen Identifikationsregeln sowie den Sicherungsmodulen zugänglich, sodass sie mit den Informationen die Zwischenfälle im Werdegang des Knotens verfolgen und abhängig von der Größe des Ringspeichers retrospektiv nachvollziehen können.
Nachdem das Ergebnis der Auswertung des `pktsnips` in der Laufzeitüberwachung vorliegt, kann durch sie entschieden werden, ob das Zustandsdelta eingeleitet wird oder der Knoten in seinem aktuellen Zustand verweilt. Hierfür wertet die Laufzeitüberwachung das Evaluierungsergebnis aus und entscheidet abschließend, ob der `pktsnip` in einer *IPC-Nachricht* an den RPL Thread weitergeleitet oder verworfen wird. Wird der `pktsnip` weitergeleitet, wird er wie zuvor von dem RPL Thread verarbeitet und leitet alle Zustandsänderungen, ausgelöst durch die enthaltenen Informationen im `pktsnip`, ein. Zum Schluss wird das Ergebnisbitfeld, das Eavluierungsergebnis und das Datenfeld für die ungeprüften Zustandsänderungen zurückgesetzt und der Thread der Laufzeitüberwachung wartet auf die nächste Nachricht von der `netapi` um seine Auswertung durchzuführen.

7.4 Evaluierung

Die Laufzeitüberwachung wurde mithilfe des `gnrc_networking examples` von RIOT Evaluiert. Für die Evaluierung wurde vorab eine RPL Topologie festgelegt und zur Laufzeit erzwungen. Hierfür wurde mittels des `gnrc_ipv6_blacklist` Moduls von RIOT eingestellt, welche Knoten miteinander kommunizieren können.
Der dadurch erzwungene DODAG besteht aus insgesamt 6 Knoten mit einer Baumtiefe von 3 Hops, vgl. Abb. 7.2. Die Knoten A und B wurden so parametriert, dass sie mit dem Root und den

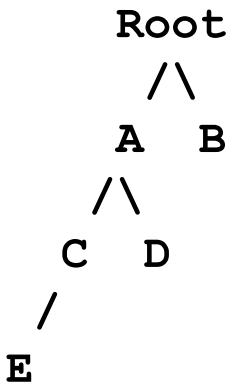

Abbildung 7.2:

Knoten C und D kommunizieren können. Die Knoten C und D wurden so parametriert, dass sie nur mit den Knoten A und E kommunizieren können. Der Knoten E wurde so parametriert, dass er mit dem Knoten C und D Kommunizieren kann.
Während der Entwicklungsphase der Implementierung wurde mittels Tests evaluiert, dass sich das Verhalten der RPL Knoten nicht durch den Einsatz der Laufzeitüberwachung ändert. Zusätzlich wurde dabei durch Tests erzwungen, dass Vorgänge innerhalb der Laufzeitüberwachung, wie die Registrierung von Identifikationsregeln oder die Zuordnung von Sicherungsmodulen das zuvor definierte Verhalten aufweisen. Die festgelegte Beispieltopologie diente hierbei als Referenz und wurde zunächst ohne die Laufzeitüberwachung auf Boards getestet. Für diese und die nachfolgenden Tests wurde das *native*[2], das *samr21-xpro*[3] und das *pba-d-01-kw2x*[4] Board eingesetzt. Dabei hat der Knoten E, in den durchgeführten Tests, C oder D als Elternknoten gewählt.
Für die Evaluierung wurde die Laufzeitüberwachung aktiviert und im festgelegten DODAG eingesetzt. Zur Evaluierung der Identifikationseinheit und den Identifikationsregeln wurde auf den Knoten überprüft, ob Ergebnisbitfeld, das Evaluierungsergebnis und die nicht überprüften Änderungen korrekt durch die Laufzeitüberwachung erfasst werden. Hierfür wurden zunächst in allgemeinen Funktionstests einzelne Durchläufe hinsichtlich der RPL Kontrollnachrichtentypen DIO, DIS, DAO, DAO-ACK und den Informationen aus dem Hop-by-Hop Paketkopf von IPv6 Paketen der Datenebene durchgeführt und mit den analog dazu Erwarteten Bits in den resultierenden 3 Bitfeldern verglichen. Es wurden dann neue Identifikationsregeln mit neuen Ergebnisbits definiert sowie vorhandene Identifikationsregeln entfernt und erneut mit den gesetzten Bits im Ergebnisbitfeld verglichen.
Zur Evaluierung der Reaktionseinheit wurden Sicherungsmodule für Tests implementiert, die bei bestimmten Konstellationen von Bits im Ergebnisbitfeld aufgerufen werden.
In allgemeinen Testfällen wurde das Verhalten einer positiven Evaluierung durch Löschen der Bits im übergebenen Bitfeld für das Evaluierungsergebnis sowie angezeigte Verstöße durch

[2] https://github.com/RIOT-OS/RIOT/wiki/Board%3A-native
[3] https://github.com/RIOT-OS/RIOT/wiki/Board%3A-SAMR21-xpro
[4] https://github.com/RIOT-OS/RIOT/wiki/Board%3A-Phytec-phyWAVE-KW22

gesetzt gebliebene Bits überprüft. Abhängig von dem jeweiligen simulierten Evaluierungsergebnis wurde überprüft, ob der Verstoß des Zustandsdeltas in der Laufzeitüberwachung protokolliert wurde. Bei einem festgestellten Verstoß durch die simulierten Überprüfungen seitens der Sicherungsmodule wurde der `pktsnip` verworfen und nicht an den RPL Thread weitergeleitet.
Zusätzlich wurden 2 explizite Testfälle definiert mit denen Fehlverhalten eines Knotens in der Topologie simuliert wurde. Hierbei wurde auf dem Knoten D das `rpl_attacker` Modul[5] aktiviert, sodass er mithilfe von Kommandos auf der RIOT shell seinen Rang und die DODAG Versionsnummer manipulieren konnte. Damit wurden reale Angriffe auf zwei wesentliche in Abschnitt 4.5 und 5 beschriebene Schwachstellen von RPL bei der Evaluierung der Reaktionseinheit simuliert. Zum Erkennen eines Angriffs mithilfe der Manipulation des annoncierten Rangs, wurde in einem Sicherungsmodul jeweils auf den Knoten A, B, C und E durch seine Implementierung festgelegt, dass Knoten D nur den Rang `3*MinHopRankIncrease` annehmen kann und nicht verbessern darf. Dabei wurde die IPv6 Adresse von Knoten D für den Testfall vorab in das Sicherungsmodul eingetragen und erzwungen, dass Knoten D die festgelegte IPv6 Adresse nutzt. Nachdem der DODAG aufgebaut war, wurde auf Knoten D der Rang mithilfe von shell-Kommandos des `rpl_attacker` Moduls manipuliert sodass er `2*MinHopRankIncrease` in seinen DIOs annonciert. Der verbesserte Rang in den annoncierten DIOs von Knoten D wurde durch die festgelegte Implementierung im Sicherungsmodul auf den übrigen Knoten erkannt und als Verstoß im Evaluierungsergebnis protokolliert. Hierbei wurde von dem jeweiligen Sicherungsmodul überprüft, dass eine im Ergebnisbitfeld angezeigte Erhöhung des Rangs nicht von der festgelegten IPv6 Adresse des Knoten D stammt. Analog dazu wurde ein Sicherungsmodul implementiert in dem das Anheben der DODAG Versionsnumer als Verstoß identifiziert wird. Hierfür wurde im Sicherungsmodul festgelegt, das die DODAG Versionsnummer nur in Kombination mit dem beitreten einer RPL Instanz und dem dazugehörigen DODAG legitim ist. Dabei wurde vom Sicherungsmodul überprüft, ob die Bits zum Anzeigen des beitreten einer RPL Instanz und dem dazugehörigen DODAG gesetzt sind, wenn die DODAG Versionsnummer erhöht wird. Ist dies nicht der Fall, bleibt das gesetzte Bit für die DODAG Versionsnummer im übergebenen Bitfeld des Evaluierungsergebnisses stehen und zeigt damit einen Verstoß an.
Die Implementierung der Laufzeitüberwachung in RIOT wurde im Hinblick auf einen minimalen Speicherbedarf entwickelt. So sind die dokumentierenden Bits eines erfassten Zustandsdeltas immer in einem Array von Bytes abgelegt, dass die kleinstmögliche Anzahl an Bytes zur Speicherung der Bits benötigt. Das Register für Identifikationsregeln sowie das der Sicherungsmodule ist auf ihre exakte zur Übersetzungszeit bekannte Anzahl begrenzt. Die ausschlaggebende Größe der Laufzeitüberwachung nimmt der Thread zur Substitution des RPL Threads ein. Er benötigt einen ausreichend großen Stack und einen Puffer für eintreffende *IPC-Nachrichten*. Werden dabei die Einstellungen des RPL Threads übernommen, belegt die Laufzeitüberwachung ca. 3kB im ROM eines RPL Knotens auf einem in den Tests eingesetzten Boards. Wird die Laufzeitüberwachung nicht auf dem Knoten eingesetzt, entsteht kein zusätzlicher Aufwand durch die Integration.

[5] https://github.com/BytesGalore/RIOT/tree/add_rpl_attacker

8 Fazit und Ausblick

Durch sie Laufzeitüberwachung ist es möglich Implementierungen zum Schutz von RPL in RIOT zu integrieren, ohne direkt in die RPL Implementierung einzugreifen. Durch die dabei entstandene entkoppelte Interaktion zwischen Routing und der Laufzeitüberwachung ist es möglich modulare Ansätze zum Schutz und zur Überwachung des Knoten zu integrieren ohne bereits bestehende Mechanismen austauschen oder erweitern zu müssen. Die Laufzeitüberwachung stellt zuverlässig die integrierten Änderungen des Zustands vom Knoten fest und protokolliert sie fortlaufend in einem Ringspeicher. Damit wird implizit eine Historie des Knotens festgehalten, sodass Statistiken über seine Zustandstransitionen direkt abgelesen werden können. Eigenen Datentypen, die in RPL Kontrollnachrichten transportiert werden, können ohne Erweiterungen von RPL durch Sicherungsmodule implementiert und registriert werden. Im Vergleich zu der *Proof of Concept Implementierung* von TRAIL in RIOT[1], können bspw. so die dort eingesetzten TVO und TVO-ACK Nachrichten ausschließlich durch die TRAIL Implementierung verwaltet werden. Das in [40] periodisch versendete Liste von vermissten Knoten im DODAG kann ebenso losgelöst von der RPL Implementierung transportiert und von einem Knoten verteilt werden. Heuristische Ansätze, die bspw. die Anzahl bestimmter ausgeführter Zustandstransitionen als Bemessungsgrundlage zum Auslösen von Gegenmaßnahmen nutzen, können direkt den Ablauf der Laufzeitüberwachung nutzen und müssen keine eigenen Überwachungsfunktionen in RPL einbringen. Durch die Trennung von RPL und den Sicherungsmodulen, können diese frei Kommunikationswege abseits des DODAGs aufbauen und nutzen. Die Laufzeitüberwachung bietet als Verwaltungsstruktur durch ihre wohldefinierten Schnittstellen und Funktionen zum Registrieren auf bevorstehende Ereignisse eine Entkoppelung verschiedene, voneinander unabhängige Sicherungsmaßnahmen gleichzeitig auf dem Knoten einzusetzen. Unabhängig von der Anzahl eingesetzter Sicherungsmaßnahmen, die auf dem Knoten zum Tragen kommen ist dadurch sichergestellt, dass jedes auf bestimmte Ereignisse registrierte Sicherungsmodul aufgerufen wird, um seine spezifische Schutzmaßnahme durchzuführen.
Die Laufzeitüberwachung wurde während der Entwicklung und Implementierung im Wesentlichen hinsichtlich der Funktionstüchtigkeit und Erfüllung der Anforderungen ihres Entwurfs evaluiert. Weiterhin kann nun genauer untersucht werden, wie viel Zusatzbelastung die Laufzeitüberwachung im Zeitverhalten sowie im Energieverbrauch des Knotens gegenüber der reinen RPL Implementierung von RIOT einbringt. Dabei sollte auch der Einsatz in Topologien untersucht werden, die sich von der definierten und sehr kontrollierten Testumgebung unterscheiden. In dem Zuge kann auch evaluiert werden, ob der Speicherverbrauch von RAM und ROM durch die Laufzeitüberwachung optimiert werden kann, bspw. durch Nachjustieren

[1] https://github.com/BytesGalore/RIOT/tree/TRAIL_SAFEST_DEMO_Legacy_migration

der Puffergrößen für den Thread der Laufzeitüberwachung. Da ausschließlich Implementierungen von Sicherungsmodulen zum Testen mit minimalem Funktionsumfang eingesetzt wurden, können diese weiterführend durch konkrete Schutzmaßnahmen, wie bspw. TRAIL, als Sicherungsmodul implementiert, erweitert und evaluiert werden.

Literaturverzeichnis

[1] Arsalan Tavakoli, Stephen Dawson-Haggerty, and P Levis, "Overview of Existing Routing Protocols for Low Power and Lossy Networks," Internet-Draft – work in progress 07, IETF, April 2009.

[2] Zach Shelby and Carsten Bormann, *6LoWPAN: The Wireless Embedded Internet*, Wiley Publishing, 1st edition, 2009.

[3] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, IETF, March 2012.

[4] J. Granjal, E. Monteiro, and J. Sá Silva, "Security for the internet of things: A survey of existing protocols and open research issues," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1294–1312, thirdquarter 2015.

[5] H. Flanagan and S. Ginoza, "RFC Style Guide," RFC 7322, IETF, September 2014.

[6] M. Dohler, T. Watteyne, T. Winter, and D. Barthel, "Routing Requirements for Urban Low-Power and Lossy Networks," RFC 5548, IETF, May 2009.

[7] K. Pister, P. Thubert, S. Dwars, and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks," RFC 5673, IETF, October 2009.

[8] A. Brandt, J. Buron, and G. Porcu, "Home Automation Routing Requirements in Low-Power and Lossy Networks," RFC 5826, IETF, April 2010.

[9] J. Martocci, P. De Mil, N. Riou, and W. Vermeylen, "Building Automation Routing Requirements in Low-Power and Lossy Networks," RFC 5867, IETF, June 2010.

[10] P. Thubert, "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)," RFC 6552, IETF, March 2012.

[11] O. Gnawali and P. Levis, "The Minimum Rank with Hysteresis Objective Function," RFC 6719, IETF, September 2012.

[12] JP. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks," RFC 6551, IETF, March 2012.

[13] T. Tsao, R. Alexander, M. Dohler, V. Daza, A. Lozano, and M. Richardson, "A Security Threat Analysis for the Routing Protocol for Low-Power and Lossy Networks (RPLs)," RFC 7416, IETF, January 2015.

[14] R. Shirey, "Internet Security Glossary," RFC 2828, IETF, May 2000.

[15] A. Barbir, S. Murphy, and Y. Yang, "Generic Threats to Routing Protocols," RFC 4593, IETF, October 2006.

[16] Chris Karlof and David Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures," *Elsevier Ad Hoc Networks*, vol. 1, no. 2–3, pp. 293–315, 2003.

[17] Carl Hartung, James Balasalle, Richard Han, Carl Hartung, James Balasalle, and Richard Han, "Node compromise in sensor networks: The need for secure systems," Tech. Rep., 2005.

[18] S. Bellovin and R. Housley, "Guidelines for Cryptographic Key Management," RFC 4107, IETF, June 2005.

[19] Wenyuan Xu, Ke Ma, W. Trappe, and Yanyong Zhang, "Jamming sensor networks: Attack and defense strategies," *Netwrk. Mag. of Global Internetwkg.*, vol. 20, no. 3, pp. 41–47, Sept. 2006.

[20] D. Whiting, R. Housley, and N. Ferguson, "Counter with CBC-MAC (CCM)," RFC 3610, IETF, September 2003.

[21] Alagumeenaakshi Muthiah and S Palaniswami, "Survey on security mechanisms for routing over 6lowpan," *International Journal of Latest Technology in Engeneering, Management and Applied Science (IJTEMAS)*, 2015.

[22] Jakob Jonsson, "On the security of ctr + cbc-mac," in *Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography*, London, UK, UK, 2003, SAC '02, pp. 76–93, Springer-Verlag.

[23] Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir, "Key recovery attacks of practical complexity on aes-256 variants with up to 10 rounds," in *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques*, Berlin, Heidelberg, 2010, EUROCRYPT'10, pp. 299–319, Springer-Verlag.

[24] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger, "Biclique cryptanalysis of the full aes," in *Proceedings of the 17th International Conference on The Theory and Application of Cryptology and Information Security*, Berlin, Heidelberg, 2011, ASIACRYPT'11, pp. 344–371, Springer-Verlag.

[25] Mihir Bellare and Phillip Rogaway, "The exact security of digital signatures-how to sign with rsa and rabin," in *Proceedings of the 15th Annual International Conference on Theory and Application of Cryptographic Techniques*, Berlin, Heidelberg, 1996, EUROCRYPT'96, pp. 399–416, Springer-Verlag.

[26] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, Florida, USA, 1996.

[27] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K. Lenstra, Emmanuel Thomé, Joppe W. Bos, Pierrick Gaudry, Alexander Kruppa, Peter L. Montgomery, Dag Arne Osvik, Herman Te Riele, Andrey Timofeev, and Paul Zimmermann, "Factorization of a 768-bit rsa modulus," in *Proceedings of the 30th Annual Conference on Advances in Cryptology*, Berlin, Heidelberg, 2010, CRYPTO'10, pp. 333–350, Springer-Verlag.

[28] Krzysztof Piotrowski, Peter Langendoerfer, and Steffen Peter, "How public key cryptography influences wireless sensor node lifetime," in *Proceedings of the Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks*, New York, NY, USA, 2006, SASN '06, pp. 169–176, ACM.

[29] Arvinderpal S. Wander, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, Washington, DC, USA, 2005, PERCOM '05, pp. 324–328, IEEE Computer Society.

[30] Nachiketh R. Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K. Jha, "Analyzing the energy consumption of security protocols," in *Proceedings of the 2003 International Symposium on Low Power Electronics and Design*, New York, NY, USA, 2003, ISLPED '03, pp. 30–35, ACM.

[31] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, IETF, August 2008.

[32] Leslie Lamport, Robert Shostak, and Marshall Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, July 1982.

[33] A. Le, J. Loo, A. Lasebae, A. Vinel, Y. Chen, and M. Chai, "The impact of rank attack on network topology of routing protocol for low-power and lossy networks," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3685–3692, Oct 2013.

[34] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schönwälder, "Using the rpl protocol for supporting passive monitoring in the internet of things," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, April 2016, pp. 366–374.

[35] Anthéa Mayzaud, Rémi Badonnel, and Isabelle Chrisment, "A Taxonomy of Attacks in RPL-based Internet of Things," *International Journal of Network Security*, vol. 18, no. 3, pp. 459–473, May 2016.

[36] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The Trickle Algorithm," RFC 6206, IETF, March 2011.

[37] A. Aris, S. F. Oktug, and S. Berna Ors Yalcin, "Rpl version number attacks: In-depth study," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, April 2016, pp. 776–779.

[38] Jorge Granjal, Edmundo Monteiro, and Jorge Sá Silva, "Security in the integration of low-power wireless sensor networks with the internet: A survey," *Ad Hoc Networks*, vol. 24, Part A, pp. 264 – 287, 2015.

[39] A. Dvir, T. Holczer, and L. Buttyan, "VeRA - Version Number and Rank Authentication in RPL," in *IEEE 8th International Conference on Mobile Adhoc and Sensor Systems (MASS)*, Oct. 2011, pp. 709–714.

[40] Kevin Weekly and Kristofer Pister, "Evaluating Sinkhole Defense Techniques in RPL Networks," in *Network Protocols (ICNP), 2012 20th IEEE International Conference on*, Nov. 2012, pp. 1–6.

[41] Martin Landsmann, Heiner Perrey, Osman Ugus, Matthias Wählisch, and Thomas C. Schmidt, "Topology Authentication in RPL," in *Proc. of the 32nd IEEE INFOCOM. Poster*, Piscataway, NJ, USA, Apr. 2013, IEEE Press.

[42] Heiner Perrey, Martin Landsmann, Osman Ugus, Matthias Wählisch, and Thomas C. Schmidt, "TRAIL: Topology Authentication in RPL," Technical Report arXiv:1312.0984, Open Archive: arXiv.org, December 2015, Initial version from December 2013.

[43] K. Chugh, L. Aboubaker, and J. Loo, "Case study of a black hole attack on lowpan-rpl," in *In: Proc. of the Sixth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*, April 2012, p. 157?162.

[44] Heiner Perrey, Martin Landsmann, Osman Ugus, Matthias Wählisch, and Thomas C. Schmidt, "TRAIL: Topology Authentication in RPL," in *Proc. of Intern. Conf. on Embedded Wireless Systems and Networks (EWSN '16)*, New York, NY, USA, Feb. 2016, pp. 59–64, ACM.

[45] Burton H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, July 1970.

[46] Linus Wallgren, Shahid Raza, and Thiemo Voigt, "Routing attacks and countermeasures in the rpl-based internet of things," *IJDSN*, vol. 2013, 2013.

[47] Shahid Raza, Linus Wallgren, and Thiemo Voigt, "Svelte: Real-time intrusion detection in the internet of things," *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2661–2674, Nov. 2013.

[48] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301, IETF, December 2005.

[49] A. Sehgal, A. Mayzaud, R. Badonnel, I. Chrisment, and J. Schönwälder, "Addressing dodag inconsistency attacks in rpl networks," in *2014 Global Information Infrastructure and Networking Symposium (GIIS)*, Sept 2014, pp. 1–8.

[50] Anthéa Mayzaud, Anuj Sehgal, Rémi Badonnel, Isabelle Chrisment, and Jürgen Schönwälder, "Mitigation of Topological Inconsistency Attacks in RPL-based Low-power Lossy

Networks," *International Journal of Network Management*, vol. 25, no. 5, pp. 320–339, 2015.

[51] R. Elz and R. Bush, "Serial Number Arithmetic," RFC 1982, IETF, August 1996.

[52] J. Hui and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams," RFC 6553, IETF, March 2012.

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 07.12.2017 Martin Landsmann