FH MÜNSTER
University of Applied Sciences

Fachbereich
Elektrotechnik und Informatik

Sicherheitsanalyse von DECT ULE

**Theoretische Analyse der Spezifikation und
Machbarkeitsstudie zur Ausnutzung gefundener Schwachstellen**

Masterarbeit

zur Erlangung des akademischen Grades
Master of Science (M. Sc.)

vorgelegt am 11. Januar 2019 von **Paul Philip Schaefer**

Matrikelnummer:	XXXXXX
Prüfer:	Prof. Dr.-Ing. Sebastian Schinzel
Zweitprüfer:	Hendrik Schwartke, M. Sc.

ETI

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht.

Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben.

Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen und ist noch nicht veröffentlicht worden. Ich bin mir bewusst, dass eine unwahre Erklärung rechtliche Folgen haben wird.

Ort, Datum Unterschrift

1 Einführung

Im Rahmen der Digitalisierung nimmt auch in den Privathaushalten die Anzahl der computergesteuerten Geräte stetig zu. Unter dem Schlagwort *Smart Home* werden konstant neue Geräte entwickelt, die eine intelligente Steuerung und Überwachung des Hauses bzw. der Wohnung versprechen.

Damit diese Geräte miteinander kommunizieren können, um beispielsweise Sensorwerte oder Kommandos auszutauschen, müssen sie miteinander verbunden werden. Eine geeignete Kommunikationsschnittstelle für besagte Smart Home Geräte zu finden ist nicht trivial, da zwei Randbedingungen beachtet werden sollten:

1. Smart Home Geräte werden häufig in bestehende Haushalte integriert, sodass eine nachträgliche Verkabelung der Geräte untereinander nur mit hohem Aufwand zu bewerkstelligen ist.
2. Der Energiebedarf der Geräte sollte so gering wie möglich sein, um auch bei batteriebetriebenen Geräten lange Betriebszeiten zu ermöglichen.

Aktuell in der IT-Landschaft verbreitete Funk-Standards wie WLAN oder Bluetooth sind für einen anderen Einsatzzweck konzipiert. Sie bieten in der Regel eine hohe Bandbreite und haben einen dementsprechend hohen Energiebedarf. Aus diesem Grund sind in den letzten Jahren einige neue Funkprotokolle für den Einsatz in Smart-Home Umgebungen definiert worden, die einen geringeren Energiebedarf haben.

Zu diesen sogenannten *Smart Home Protokollen* zählt das auf dem Funktelefon-Standard Digital Enhanced Cordless Telecommuniations (DECT) aufbauende DECT Ultra Low Energy (ULE).

Diese Arbeit beschäftigt sich mit der Sicherheit dieser Protokollvariante.

1.1 Motivation und Ziel

Die ULE Spezifikation definiert drei verschiedene Anwendungsbereiche, in denen ULE-Geräte eingesetzt werden sollen: Zum einen werden *Security applications*, zum Beispiel für Einbruchs- und Feueralarme, beschrieben, zum anderen *Global Home control and domotic* Anwendungen und zuletzt *Energy and appliances management* Szenarien. (vgl. [TS 102 939-1, Abschnitt 4.3]) Alle drei Kategorien sind sehr interessante Ziele für einen Angreifer, der beispielsweise unerlaubt die Alarmanlage deaktivieren oder die Stromzählerdaten einer Immobilie auslesen möchte. Wenn die Alarmanlage deaktiviert wird, können unbemerkt Einbrüche verübt werden und Stromzählerdaten lassen eine Rückschlüsse auf die Aktivitäten im Haus zu. Im Jahr 2012 hat eine Forschergruppe sogar zeigen können, dass man aus den Daten eines Stromzählers ermitteln kann, welches Fernsehprogramm auf dem angeschlossenen Fernseher abgespielt wurde. [DaPriM2012]

Auch die Vertraulichkeit der Daten, die über Smart Home Protokolle ausgetauscht werden, ist eine wichtige Eigenschaft eines Protokolls. Gerade in den letzten Jahren haben personenbezogene Daten durch immer besser werdende Analysemethoden stark an Nutzen für die Wirtschaft gewonnen. Daten, die im Smart Home Umfeld anfallen, sollten also möglichst gut vor unerlaubtem Mitlesen geschützt werden.

Um die Widerstandsfähigkeit des Funkprotokolls gegen Angriffe zu erhöhen, bedarf es einer fundierten Sicherheitsarchitektur. Ziel dieser Arbeit ist die Analyse von ULE hinsichtlich seiner Sicherheitseigenschaften. Anschließend wird in einer Machbarkeitsstudie evaluiert, inwiefern Angriffe auf ULE Netzwerke praktisch durchführbar sind.

Zum DECT Protokoll, auf das ULE aufbaut, wurde bereits eine ausführliche Analyse durchgeführt (siehe hierzu auch Abschnitt 1.2). In dieser Analyse sind zwei grundlegende Algorithmen, das Authentifizierungsverfahren DECT Standard Authentication Algorithm (DSAA) und der Verschlüsselungsalgorithmus DECT Standard Cipher (DSC), detailliert analysiert worden. Da diese beiden Algorithmen zumindest in Teilen gebrochen sind und DECT neuere, nach aktuellem Stand nicht gebrochene Alternativen zu diesen Algorithmen anbietet, werden diese *alten* Algorithmen in diesem Dokument nicht näher betrachtet.

1.2 Verwandte Arbeiten

Das Protokoll ULE basiert auf der DECT Technologie, die bereits 1992 in der ersten Version spezifiziert wurde. (vgl. [EN 300 175-1, Annex B]) Der ursprüngliche Einsatzzweck von DECT ist, kabellose Telefonie zu ermöglichen. Um die Vertraulichkeit von geführten Telefonaten zu gewährleisten, bietet DECT seit langer Zeit Verschlüsselungs- und Authentifizierungsalgorithmen für Verbindungen an. Die ersten Algorithmen, die hierzu definiert wurden und auch heute noch implementiert werden, sind allerdings nicht öffentlich zugänglich. Im Projekt *deDECTed* (siehe [Dedected]) wurden diese Algorithmen (DSC und DSAA) durch Reverse Engineering von bestehender DECT Hard- und Software umfangreich analysiert. In den Papern [DedectedDSAA] und [DedectedDSC] sind beide Algorithmen beschrieben, analysiert und in großen Teilen gebrochen worden.

Die deDECTed Projektgruppe hat aufbauend auf ihren Erkenntnissen neue Algorithmen für die Authentifizierung von Teilnehmern und zur Verschlüsselung für DECT Verbindungen definiert. In der Dissertation von Erik Tews [Tews2012] sind sowohl die *alten* Verschlüsselungs- und Authentifizierungsalgorithmen erläutert als auch die *neuen* Algorithmen definiert, die dann auch im DECT Standard Anwendung gefunden haben.

Auf Basis der Ergebnisse des deDECTed Projekts sind einige Implementationen für das Abhören von DECT Telefonaten entstanden. Das deDECTed Projekt selbst hat Implementationen für die *Dosch + Amand COM-ON-AIR PCMCIA-Karte* entwickelt und mit [ReDECTed] und [gr-dect2] sind inzwischen auch zwei Implementationen für Software Defined Radios verfügbar.

1.3 Aufbau der Arbeit

Diese Arbeit ist im Wesentlichen in zwei Teile unterteilt. Zunächst wird eine umfangreiche Analyse aller für ULE relevanten Spezifikationen durchgeführt. Hier wird im ersten Schritt das ULE Protokoll beschrieben und erläutert, wie ULE-Geräte kommunizieren. Nach einigen Grundlagen zum Protokoll folgt eine

Übersicht über die Sicherheitsarchitektur des Protokolls und anschließend eine Analyse einzelner *Protokollszenarien*. Ein Protokollszenario beschreibt hier eine Interaktion zweier Geräte zu einem bestimmten Zweck (zum Beispiel handelt es sich beim *Schlüsselaustausch* und bei der *Authentifizierung* jeweils um ein Protokollszenario).

Im zweiten Teil der Arbeit wird eine Machbarkeitsstudie durchgeführt, die zum Ziel hat, zu überprüfen, ob und wie die im theoretischen Teil der Arbeit gefundenen Schwachstellen tatsächlich ausgenutzt werden können. Hierzu wurde im Rahmen dieser Arbeit ein System entwickelt, mit dem grundlegende passive Angriffe auf ULE möglich sind. Insbesondere wird in diesem Teil gezeigt, welche Herausforderungen bei einem Angriff auf ULE-Geräte zu bewältigen sind.

2 Theoretische Analyse der Spezifikation

2.1 Vorgehensweise

Um eine Basis für die nachfolgende Analyse zu schaffen, wird das ULE Protokoll zunächst beschrieben. Hierzu zählt zum einen eine Beschreibung aller für ULE-Netze definierten Geräteklassen und die Erläuterung der Funktionsweise der Netzwerkschichten, die in der ULE-Kommunikation verwendet werden. Hierbei wird zunächst kein besonderer Fokus auf die Sicherheitseigenschaften des Protokolls gelegt, sondern die grundlegenden Eigenschaften des Protokolls erläutert.

Im zweiten Schritt werden alle sicherheitsrelevanten Situationen, die während des Betriebs eines ULE-Netzes auftreten, identifiziert und beschrieben. Diese Aktionen werden Protokollszenarien genannt. In diesem Teil werden insbesondere die Sicherheitseigenschaften dieser Protokollszenarien betrachtet. Nachdem ein Protokollszenario beschrieben wurde, wird dieses auf seine Sicherheit analysiert und die Vollständigkeit und Wirksamkeit der eingesetzten Sicherheitsmechanismen evaluiert.

Um eine möglichst vollständige Analyse der einzelnen Szenarien durchzuführen und die Wahrscheinlichkeit zu verringern, dass einzelne Sicherheitsgefahren übersehen werden, werden die möglichen Gefahren zunächst strukturiert modelliert und in einem *Threat Model* aufgelistet. Dieses Threat Model wird mithilfe von STRIDE aufgestellt. (vgl. [STRIDE]) STRIDE ist ein von Microsoft entwickeltes Modell von Gefahrenklassen, das man zu Hilfe nehmen kann, um Computersysteme systematisch auf Gefahren zu analysieren. Hierbei ist der Name der Methode ein Akronym der sechs Gefahrenklassen, die im Folgenden definiert werden.

Spoofing Beim *Spoofing* handelt es sich um Gefahren, bei denen ein Angreifer versucht sein Opfer zu täuschen, indem er so handelt, als hätte er eine andere Identität. Er fälscht (*spooft*) einen Kommunikationsteilnehmer und injiziert Nachrichten in dessen Namen in das Netz.

Tampering Beim *Tampering* handelt es sich um Gefahren, bei denen ein Angreifer versucht Nachrichten während der Übertragung unbemerkt zu manipulieren.

Repudiation Bei *Repudiation* handelt es sich um Gefahren, bei denen der Versand von Kommandos abgestritten wird.

Information Disclosure Bei *Information Disclosure* handelt es sich um Gefahren, bei denen ein Angreifer Daten mitliest und aus diesen Informationen gewinnt.

Denial of Service Bei *Denial of Service* handelt es sich um Gefahren, bei denen ein Dienst durch einen Angreifer – gegebenenfalls sogar nachhaltig – in seiner Funktion gestört wird.

Elevation of Privileges Bei *Elevation of Privileges* handelt es sich um Gefahren, bei denen einem Angreifer mehr Befugnisse zugesprochen werden, als ihm eigentlich zustehen.

Im Kontext einer Protokollanalyse empfiehlt sich der Einsatz der *STRIDE-per-Interaction* Variante. Diese Variante ermöglicht es, einen zusammenhängenden Ablauf, die sogenannte *interaction*, auch zusammenhängend zu betrachten und zu analysieren. Dies steht im Kontrast zur alternativen *STRIDE-per-Element* Variante, bei der jedes Element (im Protokoll-Kontext ließe sich hier beispielsweise eine einzelne Nachricht betrachten) separat betrachtet werden würde. Bei diesem Ansatz werden für jedes Tupel (*origin*, *destination*, *interaction*) Gefahren jeder der sechs Gefahrenklassen modelliert. (vgl. [STRIDE, Chapter 3, STRIDE-per-Interaction]) Bei jedem Tupel wird die Frage gestellt, welche Gefahren einer Gefahrenklasse für den Kommunikationspartner *origin* während einer bestimmten Interaktion mit dem Kommunikationspartner *destination* bestehen könnten. Da bei Netzwerkprotokollen der Sonderfall eintritt, dass in jedem Szenario, bei dem Nachrichten ausgetauscht werden, beide Kommunikationsteilnehmer sowohl *origin* einer Nachricht als auch *destination* von Nachrichten sind, wird jede STRIDE-Interaktion (also jedes Protokollszenario) aus beiden Sichtweisen analysiert, also einmal als Tupel (Initiator, Peer, Protokollszenario) und einmal als Tupel (Peer, Initiator, Protokollszenario).

Wenn alle Gefahren eines Protokollszenarios identifiziert sind, wird evaluiert, welche Mechanismen die Spezifikation anbietet, um sich vor diesen Gefahren zu schützen. An dieser Stelle gibt es verschiedene Stufen, wie die Spezifikation mit den Gefahren umgehen kann:

verpflichtendes Verfahren zur Abwehr Die Spezifikation definiert ein Verfahren zur Abwehr dieser Gefahr.

optionales Verfahren zur Abwehr Die Spezifikation definiert ein Verfahren zur Abwehr dieser Gefahr, das jedoch nicht verpflichtend einzusetzen ist.

Verfahren zur Schadensbegrenzung Es existiert ein Verfahren zur Schadensbegrenzung in der Spezifikation bzw. die Gefahr kann unter bestimmten Umständen gebannt werden. Ein Gerät kann sich allerdings nicht vollständig vor der Gefahr schützen.

vorgeschlagene Maßnahme Es existiert ein Verfahren, das zur Schadensbegrenzung beitragen würde, das allerdings nicht in der Spezifikation definiert ist. Das Gerät würde sich in der Regel weiterhin standardkonform verhalten, aber die Gefahr wird abgeschwächt.

kein Schutzmechanismus Die Gefahr kann nicht abgewehrt werden.

Nach dieser Kategorisierung wird – sofern vorhanden – der erkannte Schutzmechanismus beschrieben. Eine wichtige Einschränkung hierbei ist, dass die Protokollszenarien im Kontext von ULE-Kommunikation betrachtet werden. Das ULE Profil verpflichtet Geräte einige Sicherheitsmechanismen zu nutzen, die in anderen Profilen nur optional implementiert werden. Aus diesem Grund werden Protokollszenarien, die in anderen Profilen eher unsicher sind, gegebenenfalls als sicher eingestuft, da zumindest für ULE ausrecheinde verpflichtende Schutzmaßnahmen existieren. Falls eine Gefahr jedoch nicht vollständig abgewehrt werden kann, werden zudem noch Auswirkungen genannt, die ein Angriff haben kann, der diese Gefahr ausnutzt.

In aller Regel ist die unabhängige Betrachtung der Sicherheitseigenschaften einzelner Protokollszenarien nicht sehr realitätsnah. Wenn bestimmte Protokollszenarien nacheinander ausgeführt werden, kann das Ergebnis des ersten Protokollszenarios die nachfolgende Kommunikation beeinflussen. Wenn beispielsweise ein Schlüsselaustausch fehlschlägt, kann im Nachhinein keine verschlüsselte Verbindung aufgebaut werden. Aus diesem Grund werden gegebenenfalls auch zusammenhängende Protokollszenarien als Schutzmechanismus gegen bestimmte Gefahren gezählt.

2.2 Aufbau der Analyseteile

Jedes Protokollszenario wird in einem eigenen in sich geschlossenem Abschnitt analysiert. Hier wird zunächst das Threat Model nach STRIDE tabellarisch definiert. Da, wie oben beschrieben, jedes Protokollszenario aus beiden Perspektiven (die Perspektive des Initiators und die Perspektive des Kommunikationspartners) analysiert werden soll, werden hier zwei Tabellen erarbeitet, die jeweils die Gefahren für beide Kommunikationsteilnehmer auflisten.

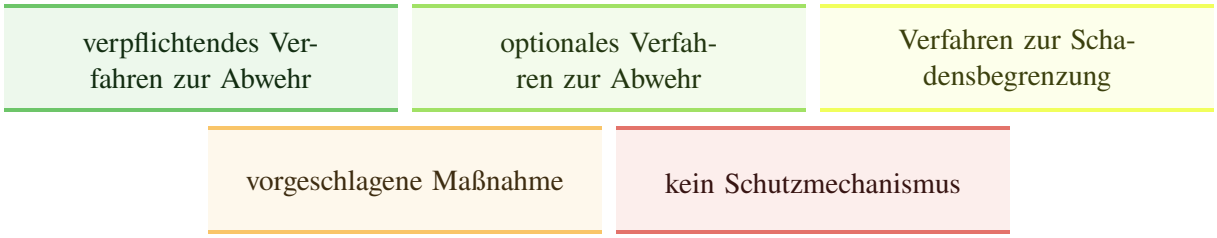
Eine Tabelle kann dabei wie folgt aussehen, wenn beispielsweise zwei Gefahren für *Spoofing* und eine Gefahr für *Information Disclosure* identifiziert wurden:

Tabelle 2.1: Threat Model aus Sicht des PP für das *Beispiel*-Szenario

Sichtweise:	PP
Protokollszenario:	Beispiel
Spoofing:	• Ein Angreifer gibt sich als legitimer Kommunikationspartner aus. [example-spoofed-fp] • Ein Angreifer führt einen Man-in-the-Middle Angriff durch. [example-mitm]
Tampering:	—
Information Disclosure:	• Ein Angreifer liest übetragene Daten mit. [example-eavesdrop-data]
Denial of Service:	—

Jede in diesen Tabellen identifizierte Gefahr ist mit einem eindeutigen Identifikator ausgezeichnet (wie [example-mitm]). Für jede dieser Gefahren wird im Anschluss separat evaluiert, inwiefern sie tatsächlich bei ULE Kommunikation eintreten kann und wie ggf. mit der Gefahr umgegangen wird. Hierzu wird für jede Gefahr eine farbkodierte Box erzeugt, dessen Farbe verdeutlicht, ob eine adäquate Maßnahme existiert, um diese Gefahr abzuwehren.

Die unter Abschnitt 2.1 „Vorgehensweise“ definierten Stufen zur Abwehr werden wie folgt in eine Farbcodierung übersetzt:



In dieser Box wird im oberen Teil die Abwehrmaßnahme zu einer Gefahr – sofern vorhanden – beschrieben und ihre Effektivität evaluiert. Auch wenn Gefahren aufgrund anderer Umstände ausgeschlossen werden können, wird dies hier diskutiert. Im unteren Teil der Box wird – sofern vorhanden – beschrieben, welche Auswirkungen ein Angriff, der diese Gefahr ausnutzt, auf das Protokollszenario hätte.

2.3 Bedrohungsmodell

Die Sicherheitsbetrachtungen in dieser Arbeit nutzen das Modell eines Angreifers, der die Möglichkeit besitzt, den Funkverkehr zwischen zwei Kommunikationsteilnehmern zu stören, mitzulesen und weitere Nachrichten zu injizieren. Dabei wird davon ausgegangen, dass sich alle Teilnehmer in der Funkreichweite des Angreifers befinden.

Der Angreifer nutzt keinen alternativen Angriffsweg, sondern ausschließlich die Funkschnittstelle, sodass Angriffe über andere Wege, wie eine mögliche IP-Schnittstelle nicht im Scope dieser Arbeit liegen.

Alle beteiligten Geräte verhalten sich vollständig konform zur Spezifikation und werden, sobald sie authentifiziert sind, als vertrauenswürdig angesehen. Sobald sich ein Gerät im Netz befindet, hat es im in dieser Arbeit verwendeten Modell das Recht alle Nachrichten zu lesen und zu senden. Dies ist analog zum Modell eines regulären Haushalts: sobald jemand Zutritt zu einem Haus hat, hat er grundsätzlich auch Zugriff auf alle Geräte innerhalb des Hauses und kann diese steuern.

Auf Basis dieses Modells fokussiert sich diese Arbeit auf die STRIDE-Gefahrenklassen *Spoofing*, *Tampering*, *Information Disclosure* und *Denial of Service*. Den beiden Gefahrenklassen *Elevation of Privileges* und *Repudiation* wird nur eine nachrangige Bedeutug zugewiesen, da – wie oben beschrieben – alle im Netz befindlichen Teilnehmer vertrauenswürdig sind und demnach alle dieselben Privilegien haben. In diesem Kontext ist eine Abstreitbarkeit von Aktionen kein relevantes Kriterium: jedem authentifizierten Gerät ist es erlaubt, jede Aktion durchführen. Schlussendlich ist es egal, welcher Teilnehmer eine bestimmte Aktion ausgelöst hat, da diese aus einem legitimen Grund ausgeführt wurde.

Ein weiterer Sonderfall bei der Analyse von Funkprotokollen ist die Gefahrenklasse des *Denial of Service*. Da ein Angreifer grundsätzlich immer in der Lage ist, die gesamte Funkkommunikation durch Jamming zu unterbinden, setzt diese Arbeit den Fokus lediglich auf die Erkennung von Denial of Service Angriffen, da ein vollständiges Verfahren zur Abwehr von Jamming nicht von einem Funkprotokoll implementiert werden kann. Wenn also bei einer Störung in der Übertragung mindestens ein Kommunikationsteilnehmer bemerkt, dass die Kommunikation nicht erfolgreich abgeschlossen wurde, wird dies bereits als *Verfahren zur Schadensbegrenzung* angesehen. Der Kommunikationsteilnehmer, der die Störung in der Übertragung bzw. das Ausbleiben einer Nachricht bemerkt, ist daraufhin nämlich in der Lage, den Schaden zu begrenzen, indem er entweder dem Nutzer anzeigt, dass die Übertragung gescheitert ist oder er die Übertragung direkt neu initiiert.

2.4 Protokollbeschreibung

ULE ist ein Funkprotokoll, das als Profil des Telekommunikationsstandards DECT definiert ist. Entsprechend verwendet ULE dieselben unteren Protokollschichten wie DECT und kann gemeinsam mit klassischen DECT Telefonen (die ihrerseits ein anderes DECT Profil verwenden) in einem Netzwerk verwendet werden. Ein ULE-Gerät übernimmt so aus Sicht der DECT Spezifikation dieselbe Rolle wie ein DECT Telefon. Viele im Folgenden erläuterten Eigenschaften von ULE gelten so gleichermaßen für alle DECT-Geräte. Aus diesem Grund werden häufig DECT-Protokollszenarien betrachtet, die allerdings konsequent im Kontext von ULE analysiert und evaluiert werden. Wenn eine Protokolleigenschaft oder ein Protokollszenario ausschließlich für ULE definiert ist, wird dies dadurch verdeutlicht, indem im Text explizit *ULE* erwähnt wird und nicht generisch auf *DECT* verwiesen wird.

2.4.1 Netzwerkteilnehmer

In DECT Netzwerken existieren in der Regel bis zu drei verschiedene Typen von Netzwerkteilnehmern, die im Folgenden definiert werden. Die Typen der Netzwerkteilnehmer sind bereits im DECT-Standard verankert, sodass die Bezeichnungen sehr stark an den ursprünglichen Einsatzzweck, die schnurlose Telefonie, angelehnt sind.

Portable Part (PP)

Ein Portable Part (PP) – also ein *Mobilteil* – ist ein Client in einem DECT Netzwerk. Hierbei handelt es sich also um ein Telefon bzw. einen Akteur oder Sensor in Smart Home Anwendungen. Jedes Portable Part (PP) hat in einem Netzwerk eine eindeutige International Portable User Identity (IPUI), worüber es adressiert werden kann.

Der Kommunikationspartner eines PP ist immer die Basisstation, das Fixed Part (FP).

Fixed Part (FP)

Eine Basisstation wird in der DECT Spezifikation Fixed Part (FP) genannt. Ähnlich wie in WLAN-Netzen sendet sie regelmäßig eine Art *Beacon*-Nachricht, um das DECT Netzwerk aufzuspannen. Ein FP beschränkt in der Regel die Kommunikation nicht auf das ULE-Profil von DECT, kündigt allerdings die Unterstützung von ULE in den Beacon-Nachrichten an.

Ein FP kann eindeutig über seinen Radio Fixed Part Identifier (RFPI) identifiziert werden.

Wireless Relay Station (WRS)

Wenn die Reichweite eines Fixed Part (FP) nicht ausreicht, kann – ähnlich wie in WLAN-Netzen – eine Art Repeater eingesetzt werden. Dieser Repeater hat im DECT-Umfeld die Bezeichnung Wireless Relay Station (WRS).

Um als *Relay Station* für die Verbindung zwischen der PP A und dem FP B zu agieren, agiert die WRS für den PP A als FP und spannt ein eigenes kleines Netzwerk auf. Dem FP B gegenüber meldet sie sich als PP an. Wenn der PP A nun die *Relay Station* nutzen möchte, muss er eine bestehende Verbindung zum FP B auf die neue Verbindung umleiten (auch *Handover* genannt).

2.4.2 Physical Layer (PHL)

Für das DECT Protokoll ist im europäischen Raum der Frequenzbereich von 1800 MHz - 1980 MHz reserviert. Die zehn Standardkanäle von DECT liegen im Frequenzband von ca. 1880 MHz - 1900 MHz und für sie ist eine Bandbreite von jeweils 1,728 MHz reserviert. Oberhalb dieses Bandes werden zusätzliche Channels angeboten, die explizit konfiguriert werden können.

Die Spezifikation definiert eine Rechenvorschrift, mit der die Frequenz eines Channels berechnet werden kann:

$$F(c) = 1897{,}344\,\text{MHz} - c \cdot 1{,}728\,\text{MHz} \qquad\qquad c \in \{\,0, 1, \ldots, 9\,\}$$

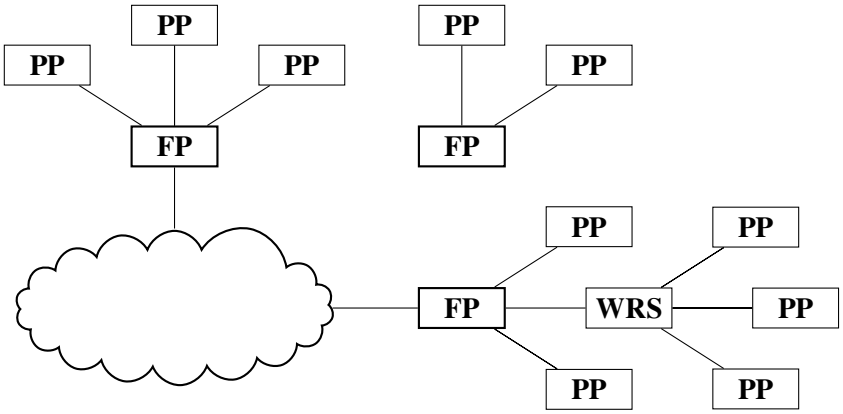


Abbildung 2.1: Beispielhafte Topologie von DECT Netzen

Auf jedem Funkkanal können mithilfe einer Time Division Multiple Access (TDMA) Struktur bis zu 12 Teilnehmer gleichzeitig mit einer Basisstation kommunizieren. Hierbei werden *Frames* mit der Länge von 10 ms in 24 *Slots* unterteilt. 16 Frames werden zu einem *Multiframe* zusammengefasst.

Jeder Slot besteht aus einem Teil, der durch Daten belegt werden kann, und einem *Guard*. Der Guard dient dazu, kurze Lücken zwischen den übertragenen Paketen zu erzeugen, um bei geringen Timing-Differenzen verschiedener Netzwerkteilnehmer die Gefahr zu verringern, dass sich die Inhalte zweier Slots vermischen. Innerhalb der Slots können *Pakete* versendet werden. DECT unterstützt eine große Anzahl verschiedener Paketlängen, wobei ULE die folgenden zwei Pakettypen unterstützt:

P32 Bei einem P32 Paket werden 424 Datensymbole versendet, das entspricht einem Slot abzüglich der Länge eines Guards.

P00 Bei einem P00 Paket werden nur 96 Datensymbole versendet, der Rest des Slots bleibt in diesem Fall leer.

DECT unterstützt diverse Modulationsmethoden, wobei ULE Gaussian Frequency Shift Keying (GFSK) mit zwei möglichen Werten pro Symbol (also ein bit pro Symbol) verwendet.

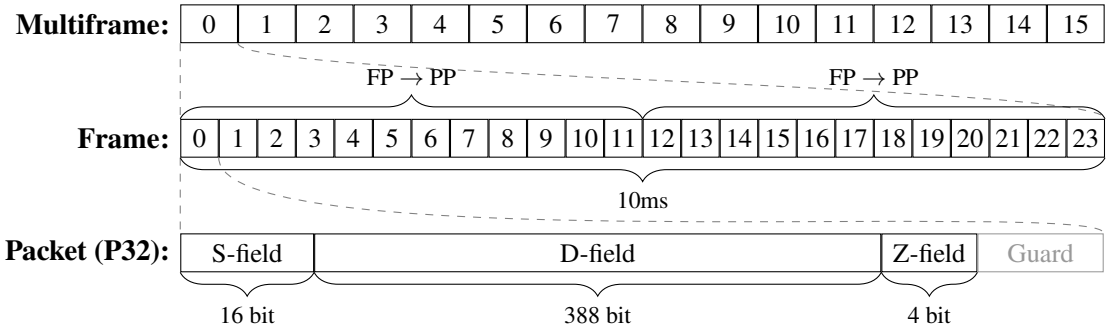


Abbildung 2.2: DECT TDMA Struktur

Jeder Slot beginnt mit einem 16-bit langen Synchronisationswort, in dem bereits kodiert ist, zu welcher Hälfte des Frames der Slot gehört. An diesem Wort synchronisieren sich die PPs. Nach dem Synchronisationswort befindet sich im D-Field der Payload. Nach dem D-Field folgt das Z-Field. Hier werden die

letzten vier Bits des D-Fields wiederholt. Dieses Feld ermöglicht einem Empfänger frühzeitig zu erkennen, wenn er nicht korrekt synchronisiert ist.

2.4.3 Media Access Control (MAC)

Innerhalb des MAC-Layers werden die verschiedenen im Physical Layer (PHL) definierten Zeitschlitze der TDMA Struktur verwaltet. Es wird definiert, dass die ersten 12 Slots eines Frames für die Kommunikation vom FP zum PP genutzt werden und die restlichen 12 Slots für die Kommunikation in die Gegenrichtung (also vom PP zum FP).

Auf dieser Struktur werden nun *Bearer* definiert. Ein Bearer ist ein Datenkanal, der zusammengehörige Zeitschlitze des PHL als eine Einheit abstrahiert. Für eine bidirektionale Verbindung zwischen einem FP und einem PP müssen zwei Zeitschlitze verwendet werden: ein Zeitschlitz aus den ersten 12 Slots eines Frames (für die Daten vom FP zum PP) und ein Zeitschlitz aus den hinteren 12 Slots eines Frames (für die Daten vom PP zum FP). Zu so einem *duplex bearer* gehören also zwei (oder mehr) Slots. Diese zwei Slots haben immer exakt einen Abstand von 12 Slots. Die Zuordnung von Datenpaketen zu MAC-Verbindungen ist also zeitabhängig.

Einen sogenannten *simplex bearer*, der nur aus einem Zeitschlitz besteht, nutzt die Basisstation um Daten an alle PPs gleichzeitig zu senden. Diesen Broadcast-Service nutzt die Basisstation beispielsweise, um einmal in jedem Multiframe den *System Information Broadcast* zu versenden, der mit WLAN-Beacon-Frames verglichen werden kann. An dieser Nachricht synchronisieren alle PPs die Multiframe-Grenzen. Da dieser Bearer nur aus einem Zeitschlitz besteht, haben die PPs nicht die Möglichkeit auf eine Broadcast-Nachricht zu antworten.

In der Abbildung 2.3 findet sich eine beispielhafte Aufteilung der Zeitschlitze in verschiedene Bearer. Hier ist in Slot 0 jeweils grau hinterlegt ein *simplex bearer* zu finden, der beispielsweise als Broadcast-Bearer zur Bekanntmachung des aktiven ULE-Netzwerkes genutzt werden kann. Grün hinterlegt ist in Slots 3 und 15 ein duplex Bearer zu finden, über den bidirektional Daten zwischen einem PP und dem FP ausgetauscht werden. Bis Frame 1 existiert noch (orange hinterlegt) ein zweiter duplex Bearer, dessen Verbindung allerdings dann beendet wird. In Slot 7 findet sich blau hinterlegt ein weiterer simplex Bearer, der eine unidirektionale Datenübertragung vom FP ermöglicht.

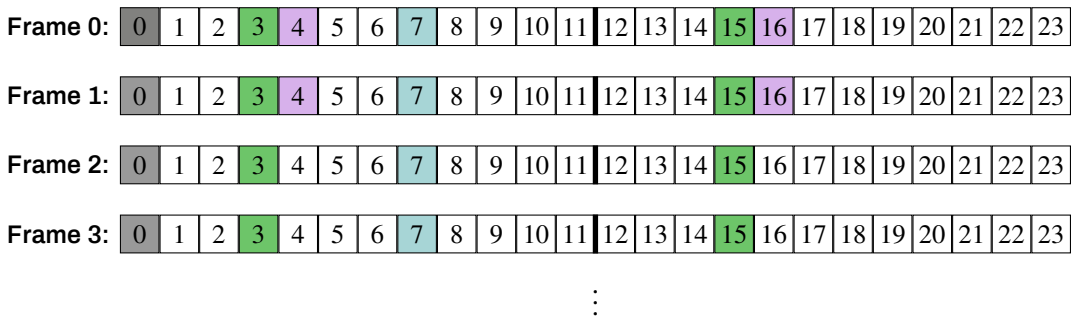


Abbildung 2.3: Exemplarische Unterteilung der TDMA-Struktur in verschiedene Bearer

Das D-Field eines Pakets wird auf dem MAC-Layer in zwei Bereiche unterteilt: das A-Field und das B-Field. Das A-Field eines Pakets wird zur Übermittlung von Metainformationen verwendet. Um beispielsweise eine neue Verbindung zu starten, nutzt ein Teilnehmer das A-Field und füllt es mit der MAC Layer

Control Message `access_request`. Im B-Field wird der Payload des MAC-Layers übertragen. Hier definiert der MAC-Layer verschiedene *Multiplexer*, die verwendet werden können, um Daten verschiedener logischer Channels innerhalb des B-Fields zu versenden.

Ein weiteres Feature des Media Access Control (MAC)-Layers ist die Verschlüsselung. Payload, der über den MAC-Layer versendet wird, kann mithilfe des Algorithmus DSC (innerhalb des deDECTed Projektes gebrochen [DedectedDSC]) oder DECT Standard Cipher #2 (DSC2) (definiert als Nachfolger von DSC und in [Tews2012] beschrieben) verschlüsselt werden. Die Spezifikation nennt die MAC Encryption ausschließlich einen *privacy mechanism*, sodass auf dieser Ebene keine Integritätssicherung vorgesehen ist.

Bevor der MAC-Layer die zu versendenden Daten an den PHL-Layer übergibt, werden die im B-Field transportierten Daten *gescrambled*. Unter *Scrambling* wird in der Funktechnik ein Vorgang beschrieben, der dazu verwendet wird, um lange Sequenzen von 0- oder 1-Werten in der Übertragung zu vermeiden. Dies vereinfacht die Rückgewinnung des Taktes auf Empfängerseite. DECT implementiert das Scrambling über eine XOR-Verknüpfung der Daten mit fest definierten, pseudozufälligen Zahlenreihen. Die Spezifikation definiert 16 verschiedene Zahlenreihen. Welche Zahlenreihe für ein bestimmtes Paket zum Scrambling verwendet wird, ist abhängig von der aktuellen Framenummer im Multiframe.

2.4.4 Data Link Control (DLC)

Der Data Link Control (DLC) Layer definiert diverse verschiedene Services zum Angebot einer nachrichtenorientierten oder streambasierten Kommunikation, wobei jedes DECT Profil eine Teilmenge dieser Services als verpflichtend ausweisen kann. In den Services wird jeweils definiert, wie Nachrichten zu puffern sind, welche Prüfsummenalgorithmen verwendet werden etc. Für ULE ist der auf dem *LU10* basierenden Dienst *LU14* einzusetzen. Des Weiteren ist die Implementation von *LU13* optional.

Die Dienste werden im Folgenden erläutert:

LU14 Enhanced Frame RELay service with CCM (EFREL-CCM) Der Dienst *LU14* bietet einen zuverlässigen, verschlüsselten und integritätsgesicherten Kommunikationskanal an. Hierzu wird Payload zunächst mit CCM verschlüsselt und integritätsgesichert und dann an den LU10-Dienst übergeben.

LU10 Enhanced Frame RELay (EFREL) service Der Dienst *LU10* ist ein nachrichtenorientierter Dienst, der Service Data Units (SDUs), also Userdaten, in Protocol Data Units (PDUs), also Nachrichten, die in ein DECT Paket passen, segmentieren und wieder zusammensetzen kann. Darüber hinaus werden übertragene PDUs bestätigt, indem Acknowledgements versendet werden.

LU13 Enhanced Frame RELay service with CRC (EFREL-CRC) Der Dienst *LU13*, bietet einen unverschlüsselten, zuverlässigen Kommunikationskanal an. Auch dieser Dienst verwendet *LU10* zur Übertragung der SDUs. Im Gegensatz zu *LU14* werden die Daten allerdings nicht zuvor verschlüsselt, sondern mit einer 16 oder 32 bit langen CRC Summe vor Übertragungsfehlern geschützt.

Bei ULE Geräten werden Applikationsdaten grundsätzlich mit LU14 versendet.

2.4.5 Network (NWK)

Oberhalb des DLC-Layers definiert DECT den Network (NWK)-Layer. Hier werden alle Services, die das DECT-Protokoll für den Application-Layer anbietet, definiert. Insbesondere werden hier einige Nachrichten und Verfahren definiert, die zur Verwaltung des Netzes und den Teilnehmern verwendet werden (sogenannte *Mobility Management*-Procedures). Hierzu gehören insbesondere auch die Nachrichten und Prozeduren zum Schlüsselaustausch, der Authentifizierung etc.

ULE verwendet gegenüber anderer DECT-Profile einen vereinfachten NWK-Layer. Bei DECT-Geräten, die andere Profile nutzen, wird ein umfangreiches Schema von *Call Control*-Procedures definiert, das die Verwaltung sowohl von paketorientierten als auch von verbindungsorientierten Kommunikationskanälen erlaubt. Bei ULE wird hingegen direkt beim Pairing ein sogenannter Permanent Virtual Circuit (PVC) aufgebaut. Dieser PVC kann im Status *Suspended* oder *Resumed* sein. Immer, wenn der NWK-Layer den Status von *Suspended* zu *Resumed* wechselt, wird eine neue MAC-Verbindung aufgebaut und wenn LU14 eingesetzt wird ein neuer Encryption-Key angefordert. Wenn kein ungenutzter Schlüssel zur Verfügung steht, wird hierzu ein neuer Schlüssel ausgetauscht.

Die in diesem Teil analysierten Protokollszenarien sind in der Regel Prozeduren, die im *Mobility Management* Teil des NWK-Layers definiert sind.

2.5 Sicherheitsarchitektur

Aufgrund der Historie als Telekommunikationsstandard definiert DECT eine umfangreiche Sicherheitsarchitektur, die genau auf den Bedarf von Telefonieanwendungen angepasst ist. So definiert DECT eine Struktur, die insbesondere mobile Geräte (deshalb auch *Portable Part*) unterstützt, die während des Einsatzes zwischen verschiedenen FP wechseln können. Diese Architektur ist so konzipiert, dass ein fremder FP keinen Zugriff auf das Master-Schlüsselmaterial, den Authentication Key des PP benötigt, um trotzdem verschlüsselte Verbindungen mit dem PP zu unterstützen.

DECT definiert grundsätzlich zwei verschiedene Arten an Algorithmen: einen Authentifizierungsalgorithmus und einen Verschlüsselungsalgorithmus.

2.5.1 A Algorithm

Zur Validierung der Authentizität der Kommunikationsteilnehmer wird ein Schema aus vier Teilprozessen verwendet, die ein Challenge-Response Verfahren implementieren. Dieses Schema wird *A Algorithm* genannt. Jeweils zwei dieser vier Teilprozesse (Algorithmen *A11* und *A12*) werden zur Authentifizierung des PP und die anderen zwei Teilprozesse (Algorithmen *A21* und *A22*) werden zur Authentifizierung des FP verwendet. Der DECT Standard definiert zwei verschiedene A Algorithms: DECT Standard Authentication Algorithm (DSAA) und DECT Standard Authentication Algorithm #2 (DSAA2) wobei auch proprietäre Alternativen durch den Standard erlaubt werden. Wie eingangs erwähnt, fokussiert sich diese Arbeit auf den Algorithmus DSAA2. Der DSAA2-Algorithmus ist der *neue* Authentifizierungsalgorithmus und verwendet zur Berechnung der 128 bit Response eine 128 bit Challenge, wobei der *alte* DSAA-Algorithmus nur mit 64 bit-Werten arbeitet.

Die vier Algorithmen, die von einem A Algorithm implementiert werden müssen, sind folgende:

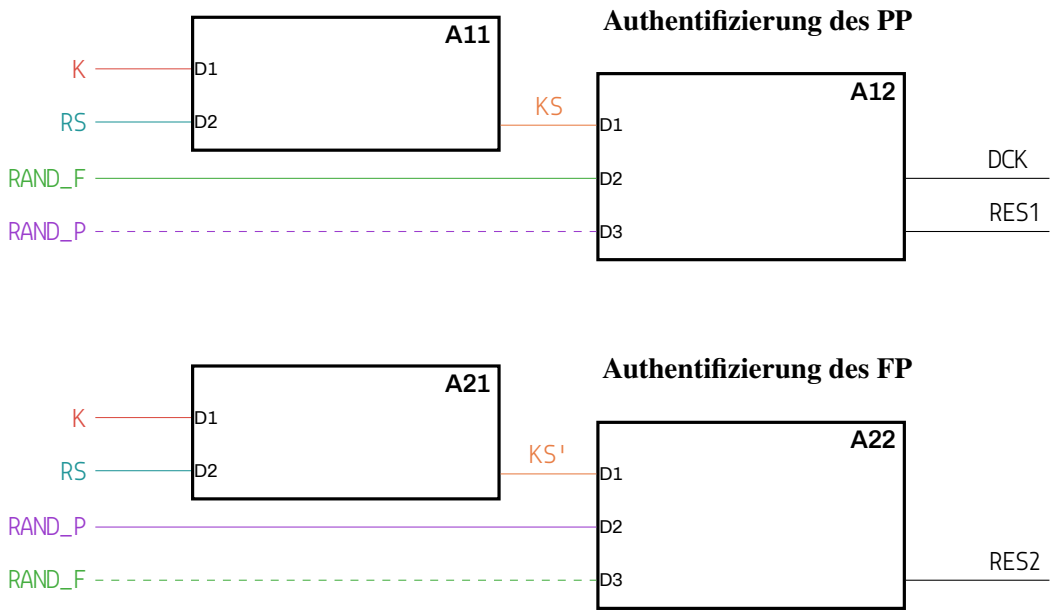


Abbildung 2.4: Schema der vier Teilprozesse der A Algorithms

A11 Der Algorithmus *A11* erzeugt aus zwei Eingaben, dem gemeinsamen geheimen Schlüssel K und einem öffentlich ausgetauschten Zufallswert *RS*, den intermediate Key *KS*. Der Wert *KS* muss nicht regelmäßig neu berechnet werden, da der Wert *RS* explizit wiederverwendet werden darf. Der Authentication Key K ist eine Art *master key* für die Verbindung zwischen FP und PP und wird nur selten erneuert.

A12 Der Algorithmus *A12* erzeugt aus dem intermediate Key *KS*, einer vom FP erzeugten Challenge *RAND_F* und einer vom PP erzeugten Challenge *RAND_P* die Werte *RES1* und DCK. Bei *RES1* handelt es sich um die Response zu der vom FP gestellten Challenge *RAND_F* und bei *DCK* um einen aus dem Challenge-Response Verfahren abgeleiteten *DCK*, der für nachfolgende Verschlüsselungen verwendet wird. Der ältere Algorithmus DSAA verwendet für den *A12* Algorithmus keinen *RAND_P*-Parameter.

A21 Der Algorithmus *A21* erzeugt, genau wie der Algorithmus *A11*, aus dem gemeinsamen geheimen Schlüssel K und dem öffentlich ausgetauschten Zufallswert *RS* einen intermediate Key KS'. Genau wie beim Algorithmus *A11* kann dieser Wert gespeichert und wiederverwendet werden.

A22 Der Algorithmus *A22* erzeugt, ähnlich wie der Algorithmus *A12*, aus dem intermediate Key KS', einer vom PP generierten Challenge *RAND_P* und einer vom FP generierten Challenge *RAND_F* die Response *RES2*. Der ältere Algorithmus DSAA verwendet für den *A22* Algorithmus keinen *RAND_F*-Parameter.

In der Abbildung 2.4 wird das Schema visualisiert. Bei übereinstimmenden Farben in der Grafik handelt es sich konzeptionell um dieselben Werte. Gestrichelte Linien beziehen sich nur auf den DSAA2-Algorithmus. Grundsätzlich gilt diese Grafik jedoch sowohl für den DSAA als auch für den DSAA2.

2.5.2 Authentication Key

Der symmetrische 128 bit lange Authentication Key K wird als gemeinsames Geheimnis zur Authentifizierung verwendet. Der Schlüssel muss dafür bei der Authentifizierung auf beiden Geräten vorliegen. Hierzu gibt es in der Regel zwei Möglichkeiten:

1. Der Authentication Key wird deterministisch aus einem kurzen (z. B. 16 bis 32 bit langen) *AC* berechnet. Der *AC* ist entweder auf dem Gerät gespeichert (ggf. vom Hersteller vorgegeben) oder wird out-of-band übertragen (z. B. manuell vom Nutzer abgelesen und auf dem anderen Gerät eingegeben).
2. Der Authentication Key wird deterministisch aus einem 128 bit langen *UAK* berechnet. Der UAK ist in jedem Fall auf dem Gerät gespeichert und kann entweder vom Hersteller vorgegeben oder während der Key Allocation (siehe Abschnitt 2.7.5) generiert worden sein.

2.5.3 DECT Standard Authentication Algorithm #2 (DSAA2)

Der DSAA2 Algorithmus definiert zwei grundlegende Algorithmen, die die vier Prozesse *A11*, *A12*, *A21* und *A22* implementieren. Diese beiden Algorithmen werden *DSAA2-1* und *DSAA2-2* genannt. Die Prozesse *A11* und *A21* werden durch denselben Algorithmus *DSAA2-1* definiert und die Prozesse *A12* und *A22* werden durch denselben Algorithmus *DSAA2-2* definiert. Der Algorithmus *DSAA2-2* erzeugt zwei Werte: einen Derived Cipher Key *DCK* und die Response zur Challenge. Der *DCK* wird im A22-Prozess jedoch nicht weiter verwendet.

DSAA2-1

Der *DSAA2-1* Algorithmus verarbeitet, wie für die Prozesse *A11* und *A12* verlangt, die Eingaben K und *RS* und erzeugt daraus den intermediate Key *KS* (*A11*) bzw. *KS′* (*A21*). Wie auch in [Tews2012] beschrieben, ist durch die Verwendung desselben Algorithmus für *A11* und *A21* der Wert für *KS* und *KS′* identisch.

Hierbei wird *KS* bzw. *KS′* wie folgt berechnet:

$$\text{AES-128}_K(\text{AES-128}_K(RS))$$

Wobei $\text{AES-128}_K(P)$ der Berechnung des AES-128 Algorithmus mit dem Schlüssel K und dem Plaintext P entspricht.

DSAA2-2

Der *DSAA2-2* Algorithmus verarbeitet drei Eingaben *D1*, *D2* und *D3* und erzeugt daraus die Werte *DCK* und *RES1* bzw. *RES2*.

Die Werte für die drei Eingaben werden wie folgt festgelegt:

Für A12:

$$D1 := KS$$
$$D2 := \textit{RAND\_F}$$
$$D3 := \textit{RAND\_P}$$

Für A22:

$$D1 := KS$$
$$D2 := \textit{RAND\_P}$$
$$D3 := \textit{RAND\_F}$$

Das Resultat *DCK* des DSAA2-2 Algorithmus berechnet sich wie folgt:

$$\text{AES-128}_{D1}\left(\text{AES-128}_{D1}\left(D2 \mid D3\right) \oplus 1\right)$$

Das Resultat *RES1* bzw. *RES2* berechnet sich wie folgt:

$$\text{AES-128}_{D1}(D2 \mid D3)$$

Wobei

- $\text{AES-128}_K(P)$ der Berechnung des AES-128 Algorithmus mit dem Schlüssel K und dem Plaintext P,
- $A \mid B$ der Konkatenation der Werte von A und B und
- $A \oplus 1$ dem Wert A mit einem Bitflip im letzten Bit

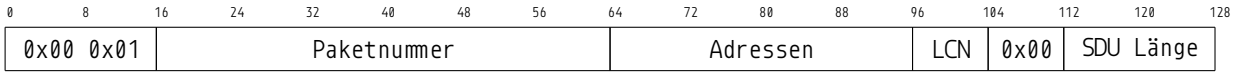
entspricht.

In der Abbildung 2.5 ist der Datenfluss der vier Prozesse visualisiert.

2.5.4 Verschlüsselung durch den DLC-Service *LU14*

Der DLC-Layer bietet seit der in 2013 erschienenen Version v2.5.1 der Spezifikation mit dem LU14-Service eine Verschlüsselung auf Basis des CCM Algorithmus an. DECT verwendet exakt den in [RFC 3610] definierten CCM Algorithmus mit AES-128.

Um zu gewährleisten, dass während der Anwendung des CCM Algorithmus ein *IV* nicht mehrfach mit demselben Schlüssel verwendet wird, definiert die Spezifikation eine Bildungsvorschrift für diesen Wert. Der *IV* wird wie folgt zusammengesetzt:



Durch den aufsteigenden Wert der Paketnummer ist sichergestellt, dass für jedes Paket ein anderer *IV* zur Verschlüsselung verwendet wird. Insbesondere definiert die DECT Spezifikation, dass ein Schlüssel, der in CCM verwendet wird, für einen Wert der *nonce* (Bytes 1-13 des *IV*) nur exakt einmal verwendet werden darf. (vgl. [EN 300 175-7, Abschnitt 6.6.2.2]) Es muss also sowohl für jede neue Verbindung (wenn die Paketnummer wieder bei 0 beginnt), als auch, wenn die CCM Sequenznummer den maximalen Wert erreicht, ein neuer Schlüssel ausgehandelt werden.

Zusätzliche Daten, die authentifiziert aber nicht verschlüsselt werden, nutzt der LU14-Service nicht.

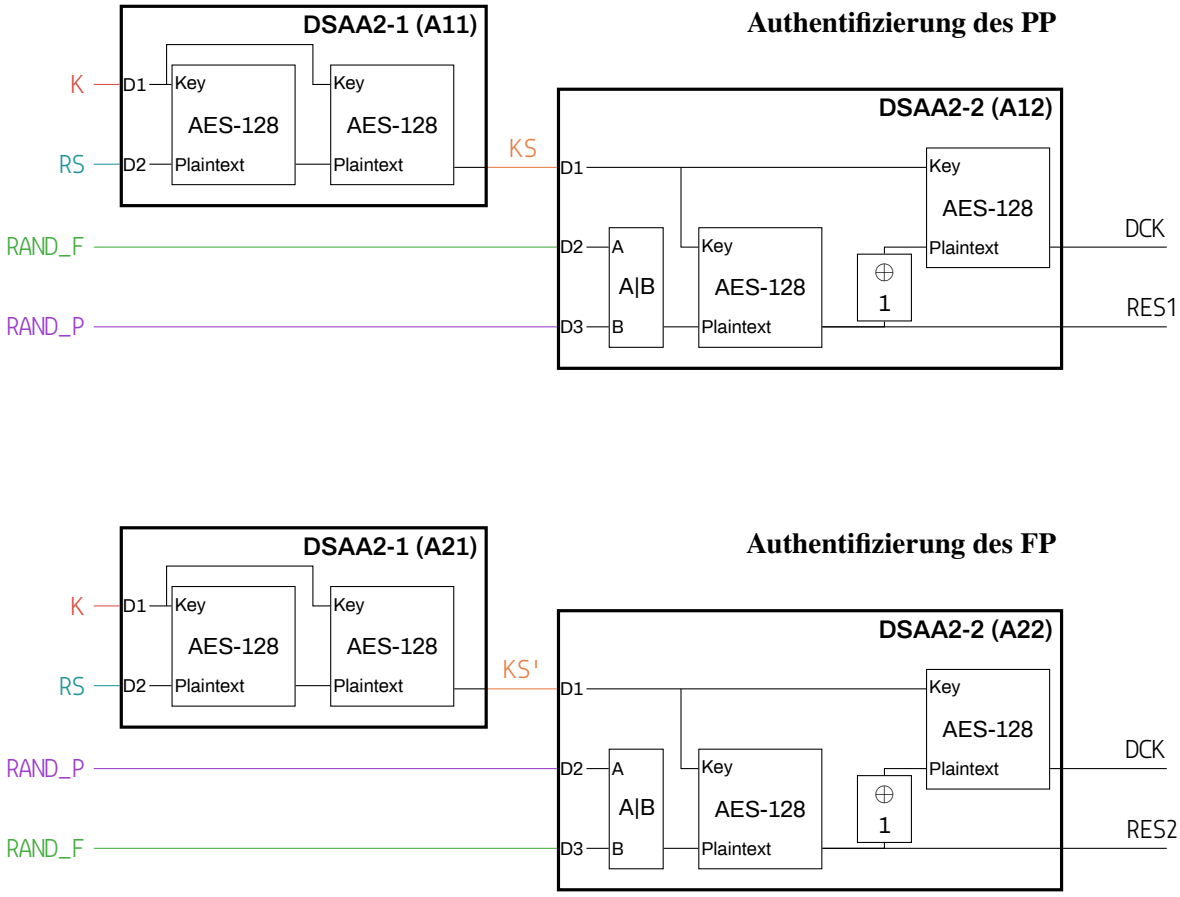


Abbildung 2.5: Schema des DSAA2 A Algorithm

2.5.5 Verschlüsselung & Roaming

DECT definiert zwei verschiedene Möglichkeiten der Verschlüsselung. Zum einen kann auf der MAC-Layer Ebene verschlüsselt werden und zum anderen kann die Verschlüsselung bereits auf dem NWK-Layer umgesetzt werden. Auch der Einsatz beider Verschlüsselungen parallel wird nicht explizit ausgeschlossen.

Die ULE Spezifikation definiert, dass alle Applikationsdaten auf dem NWK-Layer mit CCM verschlüsselt werden. Als Schlüssel wird hier, genau wie bei der MAC-Verschlüsselung, ein *DCK* als Schlüssel verwendet. Da ausgehend von einem *DCK* keine Annahmen über den Key K gemacht werden können, kann dieser DCK an weitere FPs zwecks Roaming übermittelt werden. Für neue Verbindungen wird in jedem Fall ein neuer *DCK* ausgehandelt, weshalb ein FP, der zum Roaming verwendet wurde, nicht automatisch auch nachfolgende Verbindungen entschlüsseln kann.

2.6 Übersicht der Analyse

Während der Sicherheitsanalyse der einzelnen Protokollszenarien werden einige sicherheitsrelevante Probleme im Protokoll identifiziert. Insbesondere beim Eintritt eines neuen Gerätes in ein ULE Netzwerk (die Szenarien *Obtain Access Rights*, *Key Allocation*, *Easy Pairing*) fallen Schwachstellen im Protokolldesign auf. Die Sicherstellung von Authentizität bei der Kommunikation zweier Geräte, die kein starkes gemeinsames Geheimnis besitzen, ist naturgemäß sehr komplex. Um die Authentizität und Integrität einer Kommunikationsverbindung zu überprüfen, bedarf es in einem solchen Fall immer einer dritten Partei: bei TLS handelt es sich hierbei beispielsweise um die Certificate Authorities; W-LAN delegiert diese Funktion an den Nutzer, der – ähnlich wie bei DECT – manuell in einem W-LAN Client den Pre-Shared Key konfigurieren muss. Jede Kommunikation, bei der ein Schlüssel über einen nicht authentifizierten Kanal ausgetauscht oder ausgehandelt wird, benötigt eine dritte Kontrollinstanz, die überprüfen kann, ob der übertragene Wert nicht manipuliert wurde. Diese dritte Kontrollinstanz fehlt beim *Easy Pairing*. ULE verfolgt dabei das Konzept des *Trust On First Use* – wobei erschwerend hinzukommt, dass kein Public-Key basierter Schlüsselaustausch wie Diffie-Hellman verwendet wird, sondern ein Angreifer aus den übertragenen Werten direkt den abgeleiteten Schlüssel berechnen kann.

Abgesehen von dem in ULE nicht adressierten Problem des initialen Austauschs eines gemeinsamen Geheimnisses lässt sich jedoch auch zusammenfassen, dass die Sicherheit von DECT Netzen durch die Verwendung von CCM mit AES-128, wie es bei ULE der Fall ist, deutlich verbessert wurde. Viele der hier analysierten DECT-Protokollszenarien halten der Analyse nur stand, da während der Analyse davon ausgegangen wurde, dass sie in ULE-Netzen tatsächlich nur mit einer verschlüsselten und authentifizierten Verbindung genutzt werden. Eine Verschlüsselung mit den alternativ in DECT definierten Methoden DSC2 bzw. DSC würde die Authentizität der Nachrichten nicht sicherstellen.

Die Spezifikation des ULE Protokolls ist effektiv auf acht verschiedene Dokumente aufgeteilt. Einige Protokollszenarien sind hierbei sowohl in dem Dokument definiert, das die Protokollschicht spezifiziert, für die das Protokollszenario relevant ist, als auch im Dokument „Part 7: Security features“ [EN 300 175-7]. Diese doppelte Definition von Protokollszenarien kann zu Inkonsistenzen in der Definition führen. Insbesondere bei der *Key Allocation* ist eben so eine Inkonsistenz erkannt worden: Das Dokument „Part 5: Network (NWK) layer“ [EN 300 175-5] definiert, dass die Challenge vom FP zwischenzeitlich erneuert werden soll, das Dokument „Part 7: Security features“ [EN 300 175-7] macht hierzu keine Angaben.

Im der folgenden Tabelle 2.2 werden zur Übersicht alle Gefahren aufgelistet, für die während der Sicherheitsanalyse keine beziehungsweise unzureichende Schutzmaßnahmen (also jene, die in die Kategorie *vorgeschlagene Maßnahme* bzw. *kein Schutzmechanismus* fallen) identifiziert wurden.

Tabelle 2.2: In der Sicherheitsanalyse identifizierte und nicht abgewehrte Gefahren

Szenario	**Gefahr**
Obtaining Access Rights (siehe Abschnitt 2.7.1)	Ein Angreifer ist während des *Obtain Access Rights*-Szenarios in der Lage, die präferierten A Algorithms des FP zu verändern. Wenn dies geschieht, speichert der PP diese und wird sie im Zweifel bei späteren Authentifizierungs-Szenarien nutzen – auch wenn es sich hierbei um den schwachen DSAA-Algorithmus handelt. (siehe [obtain-access-rights-downgrading-fp]) Während des *Obtain Access Rights*-Szenarios akzeptiert der FP grundsätzlich alle Verbindungen von unbekannten PPs. Dies kann ein Angreifer dazu nutzen, alle TDMA-Slots zu belegen. Dadurch kann er verhindern, dass andere (legitime) PPs Verbindungen zum FP aufbauen und ins Netz eintreten. (siehe [obtain-access-rights-occupying-tdma-dos])
Terminating Access Rights, PP initiiert (siehe Abschnitt 2.7.2)	Ein Angreifer kann die Übertragung der Austrittsanfrage eines PP stören und sich gleichzeitig dem PP gegenüber als FP ausgeben und die Anfrage bestätigen. Wenn es einem Angreifer vorher gelungen ist, den Authentication Key dieses FP/PP-Paars zu finden, kann er damit verhindern, dass dieser Authentication Key ungültig wird. (siehe [term-access-rights-pp-spoofing-fp]) Während des *Terminate Access Rights*-Szenarios sind die Nachrichten nicht vor Manipulation geschützt, weshalb ein Angreifer grundlegend dazu im Stande ist, *Reject*-Nachrichten zu *Accept*-Nachrichten zu ändern. (siehe [term-access-rights-pp-changing-to-accept])
Terminating Access Rights, FP initiiert (siehe Abschnitt 2.7.2)	*Es wurden keine Gefahren mit unzureichenden Schutzmaßnahmen identifiziert.*
Authentifizierung eines PP (siehe Abschnitt 2.7.3)	*Es wurden keine Gefahren mit unzureichenden Schutzmaßnahmen identifiziert.*
Authentifizierung eines FP (siehe Abschnitt 2.7.4)	*Es wurden keine Gefahren mit unzureichenden Schutzmaßnahmen identifiziert.*
Key Allocation (siehe Abschnitt 2.7.5)	Die *Key Allocation* läuft in der Regel mit einem 32 bit Wert als Schlüssel ab. Wenn ein Angreifer die Kommunikation während dieses Szenarios aufzeichnet, ist er im Nachhinein in der Lage, den abgeleiteten Authentication Key zu bestimmen, in dem er mithilfe von Brute-Force den genutzen 32 bit Schlüssel errät. (siehe [key-allocation-eavesdropping-challenge-response])

Szenario	Gefahr
	Ein Angreifer kann *Reject*-Nachrichten in die Kommunikation injizieren, wenn ein anderer A Algorithm verwendet werden soll, als in der *Request*-Nachricht angegeben. Durch diesen Angriff ist ein Angreifer in der Position, den genutzten A Algorithm mitzubestimmen. (siehe [key-allocation-injecting-authentication-reject-a-alg])
Easy Pairing (siehe Abschnitt 2.7.6)	Ein Angreifer kann sich als Basisstation ausgeben, da die Authentizität des FP nicht geprüft wird. Ein PP tritt so einem von einem Angreifer betriebenen Netzwerk bei. (siehe [easy-pairing-spoofing-fp]) Ein Angreifer kann während des *Easy Pairing*-Szenarios als Man-in-the-Middle agieren. Auf diese Weise verbindet sich der PP nicht direkt mit dem FP, sondern mit dem Angreifer. In dieser Position kann der Angreifer sowohl Befehle an den PP als auch an den FP schicken. Ferner können keine Nachrichten direkt vom PP und FP ausgetauscht werden, da sich beide Geräte nicht gegenseitig authentifizieren können. (siehe [easy-pairing-mitm]) Der Angreifer kann den abgeleiteten *UAK* berechnen, da in die Berechnung des *UAK* kein geheimer Wert einfließt. Mit dem *UAK* kann der Angreifer alle nachfolgenden Kommunikationen des FP mit dem PP manipulieren und mitlesen. (siehe [easy-pairing-eavesdropping-keys]) Ein Angreifer kann dem Netz beitreten, ohne dass er sich authentifizieren muss. (siehe [easy-pairing-spoofing-pp]) Ein Angreifer kann die Adresse des PP verändern, der dem Netz beitreten möchte. Der FP speichert dann die falsche Adresse und nimmt den PP nicht in das Netzwerk auf. Ein Angreifer kann so verhindern, dass dieser nach dem *Easy Pairing* mit dem FP kommunizieren kann. (siehe [easy-pairing-manipulating-ipui])
Start MAC Encryption (siehe Abschnitt 2.7.7)	*Es wurden keine Gefahren mit unzureichenden Schutzmaßnahmen identifiziert.*
Terminieren der MAC Encryption (siehe Abschnitt 2.7.8)	*Es wurden keine Gefahren mit unzureichenden Schutzmaßnahmen identifiziert.*
Schlüsselwechsel der MAC Encryption (siehe Abschnitt 2.7.9)	*Es wurden keine Gefahren mit unzureichenden Schutzmaßnahmen identifiziert.*
PVC Resumption (siehe Abschnitt 2.7.10)	*Es wurden keine Gefahren mit unzureichenden Schutzmaßnahmen identifiziert.*
PVC Suspension (siehe Abschnitt 2.7.10)	*Es wurden keine Gefahren mit unzureichenden Schutzmaßnahmen identifiziert.*
Connection Handover (siehe Abschnitt 2.7.11)	*Es wurden keine Gefahren mit unzureichenden Schutzmaßnahmen identifiziert.*

Szenario	Gefahr
Parameter Retrieval (siehe Abschnitt 2.7.12)	*Es wurden keine Gefahren mit unzureichenden Schutzmaßnahmen identifiziert.*

In den nachfolgenden Abschnitten wird jedes Protokollszenario separat betrachtet und gemäß der in Abschnitt 2.2 beschriebenen Struktur analysiert.

2.7 Analyse der Protokollszenarien

Im Folgenden werden die Protokollszenarien beschrieben, analysiert und diskutiert.

2.7.1 Obtaining Access Rights

[EN 300 175-5, Abschnitt 13.5.1]

Ein ULE-Gerät muss, um einem DECT Netzwerk beizutreten, ein Pairing-Prozess durchführen. Der Pairing Prozess kann aus zwei Teilen bestehen. Zunächst muss das Gerät das Recht im Netz zu senden erhalten (*Obtain Access Rights*). Der optionale zweite Schritt beinhaltet einen Schlüsselaustausch (*Key Allocation*).

Um das *Obtaining Access Rights*-Szenario durchführen zu können, muss der FP dies überhaupt zulassen. Im regelmäßig versendeten System Information-Broadcast (siehe Abschnitt 2.4.3 „Media Access Control (MAC)“) ist hierzu ein *Access Rights Supported*-Bit ausgewiesen, das anzeigt, ob die *Obtain Access Rights Procedure* aktuell zugelassen ist. Zur temporären Aktivierung dieses Pairing-Modus muss häufig beispielsweise ein Knopf am FP gedrückt werden.

Um das *Obtain Access Rights*-Szenario zu initiieren, sendet der PP eine `ACCESS-RIGHTS-REQUEST`-Nachricht. In dieser Nachricht übermittelt der PP seine IPUI, optional seine präferierten Verschlüsselungs- und Authentifizierungsalgorithmen und weitere Daten an den FP.

Wenn der FP die Anfrage erhält, prüft er, ob er diese bestätigen soll. Zu diesem Zeitpunkt kann der FP beispielsweise verschachtelt im *Obtain Access Rights*-Szenario den bereits erwähnten Schlüsselaustausch, die *Key Allocation* (siehe Abschnitt 2.7.5 „Key Allocation“) als sogenannte Nested Procedure durchführen. Während dieser *Key Allocation* authentifizieren sich der FP und der PP gegenseitig. Wenn dies gelingt, wird die Anfrage bestätigt, andernfalls nicht.

Um die Anfrage zu bestätigen, antwortet der FP mit einer `ACCESS-RIGHTS-ACCEPT`-Nachricht. Mit dieser Nachricht vergibt der FP auch eine neue netzinterne Adresse, den Portable Access Rights Key (PARK), an das PP. Innerhalb des Netzes wird ein PP eindeutig durch seinen IPUI und seinen PARK identifiziert. Des Weiteren sendet der FP seinerseits eine Liste von präferierten Authentifizierungs- und Verschlüsselungsalgorithmen an den PP. Theoretisch kann sich ein Gerät auch mehrfach nacheinander im Netzwerk anmelden. In diesem Fall existiert eine IPUI mit mehreren PARKs im Netz.

Wenn der FP die Anfrage ablehnt, antwortet er stattdessen mit einer `ACCESS-RIGHTS-REJECT`-Nachricht. In dieser Nachricht kann optional ein Grund hinterlegt sein, der angibt, warum die Senderechte nicht gewährt werden.

In Abbildung 2.6 ist der Nachrichtenfluss visualisiert. Teile in kursiv sind optional.

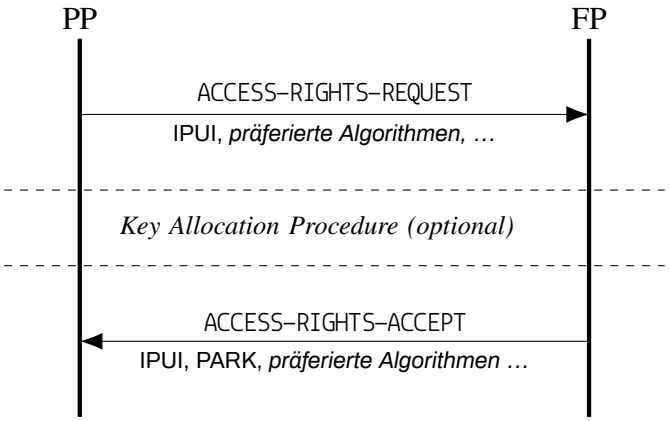


Abbildung 2.6: Nachrichtenfluss bei erfolgreichem *Obtaining Access Rights*

Threat Model

Tabelle 2.3: Threat Model aus Sicht des PP für das *Obtaining Access Rights*-Szenario

Sichtweise:	PP
Protokollszenario:	Obtaining Access Rights
Spoofing:	• Ein Angreifer spannt ein eigenes Netzwerk auf und der PP meldet sich in diesem an. [obtain-access-rights-spoofing-fp]
Tampering:	• Ein Angreifer ersetzt die vom FP festgelegten präferierten Algorithmen durch schwächere. [obtain-access-rights-downgrading-fp]
Information Disclosure:	• Ein Angreifer liest den dem PP zugewiesenen PARK mit. [obtain-access-rights-eavesdropping-park]
Denial of Service:	• Ein Angreifer kann die Kommunikation durch Jamming stören. [obtain-access-rights-jamming]

Tabelle 2.4: Threat Model aus Sicht des FP für das *Obtaining Access Rights*-Szenario

Sichtweise:	FP
Protokollszenario:	Obtaining Access Rights
Spoofing:	• Ein Angreifer täuscht vor, der PP zu sein und der FP verteilt Senderechte an den Angreifer. [obtain-access-rights-spoofing-pp]
Tampering:	• Ein Angreifer ersetzt die vom PP präferierten Algorithmen durch schwächere. [obtain-access-rights-downgrading-pp]
Information Disclosure:	—
Denial of Service:	• Ein Angreifer kann die Kommunikation durch Jamming stören. [obtain-access-rights-jamming] • Ein Angreifer kann alle TDMA-Slots belegen, während *Access Rights Supported* versendet wird, da der FP alle Verbindungen zulässt. [obtain-access-rights-occupying-tdma-dos]

Schutzmechanismen

Gefahr: [obtain-access-rights-spoofing-fp] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Es kann nicht verhindert werden, dass ein PP dem falschen Netzwerk beitritt. Wenn sich der PP jedoch in einem falschen Netzwerk befindet, wird in der weiteren Kommunikation die Authentifizierung des FP fehlschlagen, da ein Angreifer keine Kenntnis über den Authentication Key des PP hat. Deshalb wird der PP keine Kommandos von einem Angreifer ausführen und keine geheimen Daten mit ihm teilen.

Auswirkungen: Durch die nachfolgende Authentifizierung des FP nimmt der PP keine Befehle vom Angreifer entgegen. Der PP ist jedoch zunächst im falschen Netzwerk registriert. Die Registrierung muss im korrekten Netz erneut durchgeführt werden.

Gefahr: [obtain-access-rights-spoofing-pp] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Ein Angreifer kann sich als PP ausgeben, um Senderechte zu erhalten. Dies kann jedoch während des *Obtain Access Rights*-Szenario unterbunden werden, indem eine verschachtelte *Key Allocation* durchgeführt wird. Da bei ULE in jeder Übertragung die Applikationsdaten signiert und damit authentifiziert werden, wird der FP keine Nachrichten von einem Angreifer annehmen.

Auswirkungen: Ein Angreifer kann dem Netz im Zweifel beitreten, ist dann aber nicht in der Lage, Nachrichten zu versenden.

Gefahr: [obtain-access-rights-downgrading-fp] | vorgeschlagene Maßnahme

Schutzmechanismus: Wenn ein Angreifer die präferierten Algorithmen des FP ersetzt, nimmt der PP diese zunächst an. Da die in diesem Szenario ausgetauschten Werte nicht kryptographisch gesichert übertragen werden, können die Nachrichten manipuliert werden. Um einen Downgrading-Angriff zu vermeiden, sollten die hier übertragenen präferierten Algorithmen ignoriert werden und der PP sollte bei späteren Verbindungen immer zunächst versuchen, die stärksten Algorithmen zu verwenden und nur wenn diese dann abgelehnt werden auf Alternativen ausweichen.

Auswirkungen: Wenn auf die Echtheit der – möglicherweise manipulierten – Liste von Algorithmen in der `ACCESS-RIGHTS-ACCEPT`-Nachricht vertraut wird, wird möglicherweise der schwache DSAA-Algorithmus zur Authentifizierung genutzt, obwohl der FP eigentlich den DSAA2-Algorithmus bevorzugt. Bei ULE ist die Implementation des alten Algorithmus nach wie vor verpflichtend, weshalb ein Angreifer davon ausgehen kann, dass er von allen Kommunikationsteilnehmern implementiert wird.
Falls die `ACCESS-RIGHTS-ACCEPT`-Nachricht allerdings nicht manipuliert wurde, sondern es sich bei der Angabe des DSAA um eine korrekte Information handelt, scheitert die Authentifizierung zunächst, weil ein nicht unterstützter A Algorithm beim FP angefragt wird.

Gefahr: [obtain-access-rights-downgrading-pp] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Wenn ein Angreifer die präferierten Algorithmen des PP ersetzt, werden diese – ähnlich wie bei der Gefahr [obtain-access-rights-downgrading-fp] – zunächst angenommen. Da die in diesem Szenario ausgetauschten Werte nicht kryptographisch gesichert übertragen werden, können die Nachrichten manipuliert werden. Um einen Downgrading-Angriff zu vermeiden, sollten die hier übertragenen präferierten Algorithmen ignoriert werden und der FP sollte bei späteren Verbindungen immer zunächst versuchen, die stärksten Algorithmen zu verwenden. Nur wenn diese abgelehnt werden, sollte er auf Alternativen ausweichen.

Auswirkungen: Wenn die ACCESS_RIGHTS_REQUEST-Nachricht nicht manipuliert wurde und der FP andere Algorithmen nutzen möchte als der PP, würde die erste Authentifizierung zunächst scheitern, da die zu verwendenden Algorithmen abgelehnt werden würden. (vgl. Auswirkungen von [obtain-access-rights-downgrading-fp])

Gefahr: [obtain-access-rights-eavesdropping-park] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Ein Angreifer kann den PARK eines PP ohne Schwierigkeiten mitlesen und Verbindungen auf Basis dieses PARK initiieren. In ULE-Netzen wird allerdings bei jeder Verbindung zum FP die Authentizität des PP geprüft, indem der FP das *Authentifizierung eine PP*-Szenario durchführt. Ein Angreifer ist nicht in der Lage sich korrekt als PP zu authentifizieren, auch wenn er den PARK des PP kennt.

Auswirkungen: Ein Angreifer kann Verbindungen zum FP initiieren und ggf. damit für kurze Zeit die vorhandenen TDMA-Ressourcen verringern.

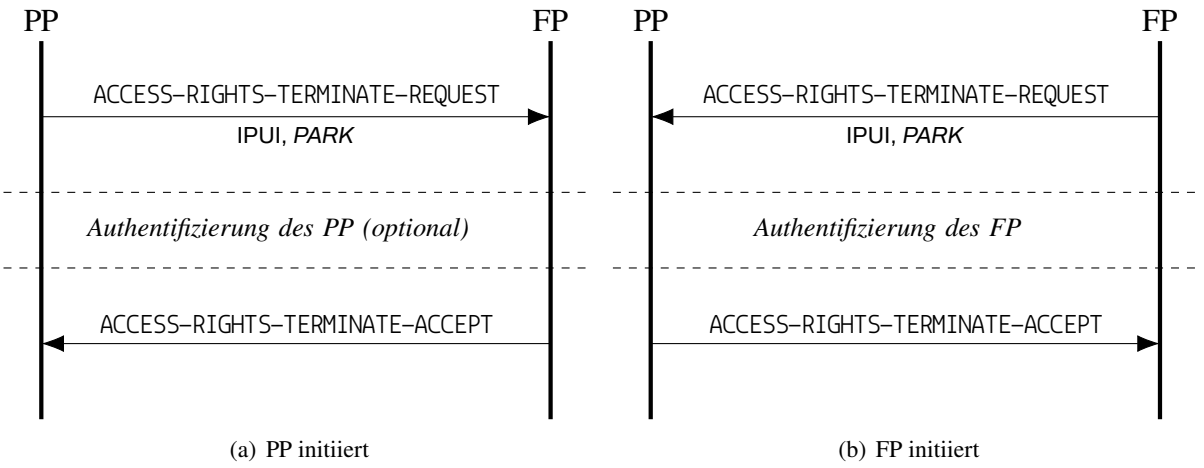
Gefahr: [obtain-access-rights-jamming] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Wenn die Datenübertragung gestört ist, ist eine Anwendung in der Lage den Verlust von denjenigen Nachrichten zu erkennen, auf die eine Antwort erwartet wird. Im *Obtain Access Rights*-Szenario kann der PP also prüfen, ob eine Antwort auf die ACCESS_RIGHTS_REQUEST Nachricht ankommt. Wenn keine Antwort ankommt, kann er darauf schließen, dass entweder die Übertragung der ACCESS_RIGHTS_REQUEST-Nachricht oder die Übertragung der Antwort auf diese Nachricht nicht erfolgreich war und ggf. gestört wurde.

Auswirkungen: Die Kommunikation ist gestört und muss erneut gestartet werden.

Gefahr: [obtain-access-rights-occupying-tdma-dos] | kein Schutzmechanismus

Auswirkungen: Wenn alle TDMA Slots belegt sind, können keine weiteren Verbindungen aufgebaut werden. Zeitweise ist die Kommunikation auch für bereits mit dem FP verbundene PPs gestört.



(a) PP initiiert (b) FP initiiert

Abbildung 2.7: Nachrichtenfluss bei erfolgreichem *Terminating Access Rights*

2.7.2 Terminating Access Rights

[EN 300 175-5, Abschnitt 13.5.2]

Um ein PP aus einem DECT Netzwerk zu entfernen, wird das *Terminating Access Rights*-Szenario definiert. Nach Durchführung dieses Szenarios kann der PP keine Verbindungen mehr im Netzwerk aufbauen und es wird kein Schlüsselmaterial mehr auf dem Gerät gespeichert.

Dieses Szenario kann sowohl vom PP als auch vom FP initiiert werden. Im Folgenden werden beide Varianten beschrieben.

Terminating Access Rights, PP initiiert

Um aus dem Netzwerk auszutreten, sendet ein PP die ACCESS-RIGHTS-TERMINATE-REQUEST-Nachricht an den FP. In dieser Nachricht wird in jedem Fall die IPUI des PP übertragen, optional ist zusätzlich die Angabe des PARK.

Wenn der FP diese Nachricht empfängt, kann er in einer Nested Procedure die Authentifizierung des anfragenden PPs verlangen (siehe Abschnitt 2.7.3 „Authentifizierung eines PP“). Wenn diese Authentifizierung fehlschlägt, wird die Austrittsanfrage des PP abgelehnt und dies mit einer ACCESS-RIGHTS-TERMINATE-REJECT-Nachricht quittiert.

Nach erfolgreicher Authentifizierung löscht der FP die Senderechte des angegebenen PP aus seiner Registratur. Wenn in der ACCESS-RIGHTS=TERMINATE-REQUEST-Nachricht eine PARK angegeben ist, löscht er nur die zum PARK gehörigen Werte; wenn kein PARK angegeben ist, werden alle mit dem IPUI assoziierten Daten (und somit auch alle zur IPUI gehörenden PARKs) gelöscht. Nach erfolgreicher Löschung der Daten antwortet der FP mit der ACCESS-RIGHTS-TERMINATE-ACCEPT-Nachricht.

Bei Empfang dieser Nachricht löscht der PP dieselben Daten, die auch der FP zuvor gelöscht hat: entweder die zum versandten PARK gehörigen Daten oder alle mit dem IPUI assoziierten Daten.

Threat Model

Tabelle 2.5: Threat Model aus Sicht des FP für das PP initiierte *Terminate Access Rights*-Szenario

Sichtweise:	PP
Protokollszenario:	Terminating Access Rights, PP initiiert
Spoofing:	• Ein Angreifer stört die Übertragung des ACCESS-RIGHTS-TERMINATE-REQUEST und antwortet im Namen des FP mit der ACCESS-RIGHTS-TERMINATE-ACCEPT-Nachricht. [term-access-rights-pp-spoofing-fp]
Tampering:	• Ein Angreifer verändert eine ACCESS-RIGHTS-TERMINATE-REJECT Nachricht in eine ACCESS-RIGHTS-TERMINATE-ACCEPT-Nachricht. Der PP akzeptiert dann keine Verbindungen des FP mehr, obwohl der FP die Anfrage eigentlich abgelehnt hat. [term-access-rights-pp-changing-to-accept]
Information Disclosure:	—
Denial of Service:	• Ein Angreifer stört die Kommunikation durch Jamming. [term-access-rights-pp-jamming]

Tabelle 2.6: Threat Model aus Sicht des FP für das PP initiierte *Terminate Access Rights*-Szenario

Sichtweise:	FP
Protokollszenario:	Terminating Access Rights, PP initiiert
Spoofing:	• Ein Angreifer versendet im Namen eines PPs eine ACCESS-RIGHTS-TERMINATE-REQUEST-Nachricht. [term-access-rights-pp-spoofing-pp]
Tampering:	• Ein Angreifer manipuliert den IPUI bzw. PARK des austretenden PPs in der ACCESS-RIGHTS-TERMINATE-REQUEST-Nachricht. [term-access-rights-pp-manipulating-ipui-park]
Information Disclosure:	—
Denial of Service:	• Ein Angreifer stört die Kommunikation durch Jamming. [term-access-rights-pp-jamming]

Schutzmechanismen

Gefahr: [term-access-rights-pp-spoofing-fp] | vorgeschlagene Maßnahme

Schutzmechanismus: Wenn sich ein Angreifer als FP ausgibt und in seinem Namen eine ACCESS-RIGHTS-TERMINATE-ACCEPT-Nachricht sendet, geht ein PP zunächst davon aus, dass die Prozedur erfolgreich war. Ein PP könnte jedoch nach Durchführung dieser Prozedur testen, ob eine neue Verbindung zum FP augebaut werden kann. Ist dies nicht der Fall, so ist die Prozedur entweder erfolgreich gewesen oder der Angreifer hat auch den Test-Verbindungsaufbau manipuliert.

Auswirkungen: Ein Angreifer kann durch Manipulation dieser Prozedur erreichen, dass ein Authentication Key beim FP seine Gültigkeit behält. Wenn ein Angreifer den Authentication Key einer

FP/PP Verbindung kennt (z. B. hat er diesen durch Brute-Force erraten), kann er verhindern, dass diesem Schlüssel die Rechte entzogen werden.

Gefahr: [term-access-rights-pp-spoofing-pp] | optionales Verfahren zur Abwehr

Schutzmechanismus: Um das Injizieren einer ACCESS-RIGHTS-TERMINATE-REQUEST im Namen des PP zu verhindern, kann ein FP die *Authentifizierung eines PP*-Prozedur ausführen, bevor er die Anfrage ausführt und daraufhin bestätigt. Ein Angreifer ist nicht in der Lage sich ordnungsgemäß als PP zu authentifizieren, weshalb der FP in dem Fall die Anfrage ablehnen würde.

Auswirkungen: Wenn der FP keine Authentifizierung während des *Terminate Access Rights*-Szenario durchführt, ist ein Angreifer in der Lage einen PP aus dem Netzwerk zu entfernen.

Gefahr: [term-access-rights-pp-changing-to-accept] | kein Schutzmechanismus

Auswirkungen: Der PP tritt aus dem Netzwerk aus, obwohl der FP die Austrittsanfrage abgelehnt hat.

Gefahr: [term-access-rights-pp-manipulating-ipui-park] | optionales Verfahren zur Abwehr

Schutzmechanismus: Bei einer Veränderung des IPUI bzw. des PARK in der ACCESS-RIGHTS-TERMINATE-REQUEST-Nachricht können folgende zwei Situationen eintreten:

- Der FP findet zum angegebenen IPUI bzw. PARK keinen registrierten PP und das Szenario scheitert daraufhin.
- Der FP findet zum angegebenen IPUI bzw. PARK einen registrierten PP, der vom eigentlichen Kommunikationspartner abweicht. In diesem Fall wird die optionale Authentifizierung des PP fehlschlagen, da der FP davon ausgeht mit einem anderen PP zu kommunizieren.

Auswirkungen: Das Szenario scheitert.

Gefahr: [term-access-rights-pp-jamming] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Wenn ein Angreifer die Kommunikation durch Jamming stört, können keine Nachrichten übermittelt werden. Der PP tritt infolgedessen nicht aus dem Netz aus. Da er allerdings auch keine Antwort auf die ACCESS-RIGHTS-TERMINATE-REQUEST-Nachricht erhält, kann er die Störung in der Übertragung erkennen und das Szenario erneut initiieren.

Auswirkungen: Der PP tritt nicht aus dem Netz aus und muss das Szenario erneut initiieren.

Terminating Access Rights, FP initiiert

Genau wie bei der PP initiierten Variante dieses Szenarios, initiiert der FP das *Terminating Access Rights*-Szenario, indem er die ACCESS-RIGHTS-TERMINATE-REQUEST-Nachricht an den PP versendet. Analog definiert der FP in dieser Nachricht, welche Senderechte exakt entzogen werden sollen: entweder gibt er nur die IPUI an, oder er definiert einen einzelnen PARK, für den die Senderechte entzogen werden sollen.

Wenn der PP diese Nachricht empfängt, sollte[1] dieser die Authentizität des FP überprüfen, indem das *Authentifizierung eines FP*-Szenario ausgeführt wird (siehe Abschnitt 2.7.4 „Authentifizierung eines FP“). Wenn diese Authentifizierung fehlschlägt, beendet der PP das Szenario, indem er mit einer ACCESS-RIGHTS-TERMINATE-REJECT-Nachricht antwortet.

Ist die Authentifizierung des FP erfolgreich, löscht der PP die zur angegebenen Identität gehörigen Daten und antwortet mit der ACCESS-RIGHTS-TERMINATE-ACCEPT-Nachricht.

Wenn der FP die ACCESS-RIGHTS-TERMINATE-ACCEPT-Nachricht empfängt, betrachtet er das Szenario als erfolgreich beendet.

Threat Model

Tabelle 2.7: Threat Model aus Sicht des PP für das FP initiierte *Terminate Access Rights*-Szenario

Sichtweise:	PP
Protokollszenario:	Terminating Access Rights, FP initiiert
Spoofing:	• Ein Angreifer entzieht die Senderechte eines beliebigen PP, indem er sich als FP ausgibt. [term-access-rights-fp-spoofing-fp]
Tampering:	• Ein Angreifer manipuliert den IPUI bzw. PARK des austretenden PPs in der ACCESS-RIGHTS-TERMINATE-REQUEST-Nachricht. [term-access-rights-fp-manipulating-ipui-park]
Information Disclosure:	—
Denial of Service:	• Ein Angreifer stört die Kommunikation durch Jamming. [term-access-rights-fp-jamming]

[1] siehe im Vergleich die PP initiierte Variante: hier wird definiert, dass der FP die Authenfizierung ausführen *kann*, nicht zwangsläufig *soll*

Tabelle 2.8: Threat Model aus Sicht des FP für das FP initiierte *Terminate Access Rights*-Szenario

Sichtweise:	FP
Protokollszenario:	Terminating Access Rights, FP initiiert
Spoofing:	—
Tampering:	• Ein Angreifer verändert eine ACCESS-RIGHTS-TERMINATE-REJECT Nachricht in eine ACCESS-RIGHTS-TERMINATE-ACCEPT-Nachricht. Der FP akzeptiert dann keine Verbindungen des PP mehr, obwohl der PP die Anfrage eigentlich abgelehnt hat. [term-access-rights-fp-changing-to-accept]
Information Disclosure:	—
Denial of Service:	• Ein Angreifer stört die Kommunikation durch Jamming. [term-access-rights-fp-jamming]

Schutzmechanismen

Gefahr: [term-access-rights-fp-spoofing-fp] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Um zu verhindern, dass der FP gespooft wird, sollte ein PP den FP authentifizieren, bevor er die Anfrage annimmmt und seine Daten löscht. Ein Angreifer kann sich nicht ordnungsgemäß als FP authentifizieren, weshalb der PP in dem Fall die Anfrage ablehnen würde.

Auswirkungen: Das Szenario scheitert und muss neu initiiert werden.

Gefahr: [term-access-rights-fp-manipulating-ipui-park] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Da DECT grundsätzlich verbindungsorientiert kommuniziert, wird der PP mindestens bei einer Manipulation des IPUI bemerken, dass die Nachricht inkorrekt ist. Der PP wird deshalb mit der ACCESS-RIGHTS-TERMINATE-REJECT-Nachricht antworten und das Szenario abbrechen. Auch wenn der Angreifer den PARK auf einen anderen PARK des PP ändert, wird das Szneario scheitern. In diesem Fall wird der PP das *Authentifizierung eines FP*-Szenario im Namen des fälschlich empfangenen PARK durchführen und der FP erwartet die Authentifizierung im Namen eines anderen PARK.

Auswirkungen: Das Szenario scheitert und muss neu initiiert werden.

Gefahr: [term-access-rights-fp-changing-to-accept] | kein Schutzmechanismus

Auswirkungen: Der PP tritt aus dem Netzwerk aus, obwohl er die Austrittsanfrage abgelehnt hat.

Gefahr: [term-access-rights-fp-jamming]	Verfahren zur Schadensbegrenzung

Schutzmechanismus: Wenn die Kommunikation gestört ist, tritt der PP nicht aus dem Netz aus, da er die Austrittsanfrage nicht erhält. Da der FP keine Antwort auf die ACCESS_RIGHTS_TERMINATE_REQUEST-Nachricht erhält, kann er die Störung in der Übertragung erkennen und das Szneario erneut initiieren.

Auswirkungen: Der PP tritt nicht aus dem Netz aus und der FP muss das Szenario erneut initiieren.

2.7.3 Authentifizierung eines PP

[EN 300 175-5, Abschnitt 13.3.1], [EN 300 175-7, Abschnitt 6.3.3.3]

Ein FP kann einen PP authentifizieren, indem er die Prozedur *Authentication of PP* durchführt. Diese Prozedur ist ein Challenge-Response-Verfahren, bei dem der PP nachweist, den Authentication Key zu kennen. Hierzu werden die beiden Teilprozesse *A11* und *A12* des A Algorithm genutzt (siehe Abschnitt 2.5.1 „A Algorithm“). Neben der Prüfung der Authentizität des PP wird durch diese Prozedur ein neues gemeinsames Geheimnis, der *DCK* berechnet, welches zur Verschlüsselung der nachfolgenden Kommunikation verwendet werden kann. Nicht zuletzt um einen neuen *DCK* zu erzeugen, wird üblicherweise, direkt nachdem eine neue Verbindung aufgebaut wird, eine Authentifizierung beider Kommunikationspartner durchgeführt.

Zur Initiierung dieses Szenarios sendet der FP eine AUTHENTICATION_REQUEST-Nachricht an den PP. In dieser Nachricht teilt der FP dem PP mit, welchen A Algorithm er verwenden möchte und definiert den Wert für *RS* und die Challenge *RAND_F*. Bei dem Wert *RAND_F* handelt es sich um einen vom FP zufällig generierten 64 bit Wert und bei *RS* um einen zufällig generierten 128 bit Wert. Um den Aufwand späterer Durchführungen des Authentifizierungsalgorithmus zu verringern, darf der *RS*-Wert wiederverwendet werden.

Bei Empfang der Authentifizierungsanfrage prüft der PP, ob er die Anfrage verarbeiten soll. Hierzu kann er optional beispielsweise die Prozedur *Authentication of FP* als Nested Procedure durchführen. Wenn diese Prozedur gelingt, würde der PP die Anfrage bestätigen, andernfalls ablehnen. Um die Anfrage abzulehnen, sendet der PP eine AUTHENTICATION_REJECT-Nachricht an den FP. Optional kann in dieser Nachricht ein *REJECT-REASON* angegeben werden.

Wenn die Anfrage bestätigt werden soll, berechnet der PP zunächst die Response zur vom FP gestellten Challenge *RAND_F*. Hierzu führt er die Prozesse *A11* und *A12* des A Algorithm aus. Dabei prüft er zunächst, ob er bereits einen passenden Wert für *KS* gespeichert hat (also, ob der *RS*-Wert wiederverwendet wird); wenn dies nicht der Fall ist, berechnet der PP den *A11* Prozess: $KS :=$ AES-128$_K$(AES-128$_K$(RS)).

Wenn der PP den Wert von *KS* bestimmt hat, erzeugt er einen eigenen Teil der Challenge *RAND_P* mit einer Länge von 64 bit. Mit den drei Werten *KS*, *RAND_F* und *RAND_P* kann der PP nun den letzten Prozess *A12* der *Authentifizierung eines PP* durchführen und damit die Werte *RES1* und *DCK* berechnen:

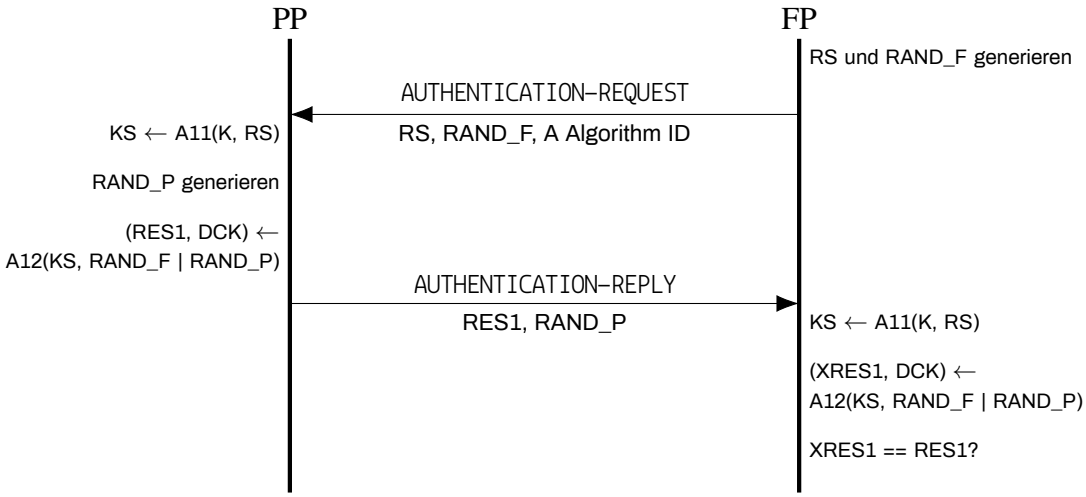


Abbildung 2.8: Nachrichtenfluss bei erfolgreicher *Authentifizierung eines PP*

$$\begin{aligned} \textit{RES1} &:= \text{AES-128}_{KS}(\textit{RAND\_F} \mid \textit{RAND\_P}) \\ \textit{DCK} &:= \text{AES-128}_{KS}(\text{AES-128}_{KS}(\textit{RAND\_F} \mid \textit{RAND\_P}) \oplus 1) \end{aligned}$$

Die Notation ist hier analog zu Abschnitt 2.5.3 „DECT Standard Authentication Algorithm #2 (DSAA2)“.

Das Ergebnis *RES1* wird hier als Response zur Challenge *RAND_F* | *RAND_P* verwendet. Zusammen mit dem Wert *RAND_P* schickt der PP diesen Wert in der Nachricht AUTHENTICATION–REPLY zurück zum FP. Der FP ist bei Erhalt der AUTHENTICATION–REPLY-Nachricht in der Lage, die empfangenen Werte zu verifizieren, da er alle benötigten Parameter der Algorithmen kennt. Der FP berechnet den Wert *XRES1* (*eXpected RES1*) aus allen ihm bekannten Parametern und vergleicht diesen mit dem empfangenen Wert von *RES1*. Stimmen diese beiden Werte überein, wird die Authentizität des PP als validiert betrachtet.

Aus der Berechnung des A12-Prozesses resultiert nach dem Wert *RES1* bzw. *XRES1* auch der *DCK*, der geheim bleibt. Wenn beide Partner denselben Wert für *RES1*/*XRES1* berechnet haben, wird davon ausgegangen, dass auch der *DCK* identisch ist. Dieser Wert kann im weiteren Kommunikationsverlauf zur symmetrischen Verschlüsselung verwendet werden.

Der Nachrichtenfluss einer erfolgreichen Durchführung dieses Szenarios ist in Abbildung 2.8 dargestellt.

Beim Einsatz des veralteten DSAA-Algorithmus ergibt sich eine leichte Variation in diesem Szenario: Hier erzeugt der PP keinen eigenen Teil *RAND_P* der Challenge und die Länge des Zufallswertes *RS* beträgt lediglich 64 bit.

Threat Model

Tabelle 2.9: Threat Model aus Sicht des PP für das *Authentifizierung eines PP*-Szenario

Sichtweise:	PP
Protokollszenario:	Authentifizierung eines PP
Spoofing:	—
Tampering:	• Ein Angreifer verändert den vom FP festgelegten A Algorithm. [auth-pp-downgrading]
Information Disclosure:	• Ein Angreifer liest mit *RS*, *RAND_F* und *RAND_P* Teile des Schlüsselmaterials mit. [auth-pp-eavesdropping-key-material]
Denial of Service:	• Ein Angreifer manipuliert einen Wert, um die Authentifizierung scheitern zu lassen. [auth-pp-manipulating-value-dos] • Ein Angreifer stört die Kommunikation durch Jamming. [auth-pp-jamming]

Tabelle 2.10: Threat Model aus Sicht des FP für das *Authentifizierung eines PP*-Szenario

Sichtweise:	FP
Protokollszenario:	Authentifizierung eines PP
Spoofing:	• Ein Angreifer gibt sich als der zu authentifizierende PP aus und der FP authentifiziert daraufhin einen Angreifer. [auth-pp-spoofing-pp]
Tampering:	—
Information Disclosure:	• Ein Angreifer liest eine valide Response zu einer Challenge mit. [auth-pp-eavesdropping-challenge-response]
Denial of Service:	• Ein Angreifer manipuliert einen Wert, um die Authentifizierung scheitern zu lassen. [auth-pp-manipulating-value-dos] • Ein Angreifer stört die Kommunikation durch Jamming. [auth-pp-jamming]

Schutzmechanismen

Gefahr: [auth-pp-spoofing-pp]	verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Um sich als legitimer PP ausgeben und die Authentifizierung korrekt durchführen zu können, benötigt der Angreifer Kenntnis über den Wert des Authentication Key. Die Authentifizierung eines Angreifers scheitert, weil er diesen Wert nicht kennt.

Auswirkungen: Die Authentifizierung scheitert und muss neu initiiert werden.

Gefahr: [auth-pp-downgrading]	Verfahren zur Schadensbegrenzung

Schutzmechanismus: Wenn ein Angreifer den A Algorithm in der AUTHENTICATION-REQUEST-Nachricht ändert, kann der PP diesen ablehnen. Dies tut er, indem er mit einer AUTHENTICATION-REJECT-Nachricht antwortet.

Wenn der PP den manipulierten A Algorithm verwendet, wird die Authentifizierung spätestens auf Seiten des FP scheitern, da er mit dem ursprünglich in der Nachricht kodierten A Algorithm einen anderen Wert für *RES1* berechnet als der PP.

Ein schwacher A Algorithm (DSAA) sollte allerdings trotzdem zunächst abgewiesen werden. Gegebenenfalls kann ein Angreifer nämlich aus einem mit einem schwachen Algorithmus berechneten *RES1* Rückschlüsse auf den verwendeten Authentication Key oder den erhaltenen *DCK* ziehen.

Auswirkungen: Wenn der Angreifer den A Algorithm manipuliert, scheitert die Authentifizierung. Sie muss neu initiiert werden.

Gefahr: [auth-pp-eavesdropping-key-material]	verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Bei einem nicht gebrochenen A Algorithm kann ein Angreifer aus den Werten *RS*, *RAND_F* und *RAND_P* keine weiteren Informationen ableiten.

Gefahr: [auth-pp-eavesdropping-challenge-response]	verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Bei einem nicht gebrochenen A Algorithm kann ein Angreifer aus einer Response zu einer Challenge keine weiteren Informationen ableiten.

Gefahr: [auth-pp-manipulating-value-dos]	Verfahren zur Schadensbegrenzung

Schutzmechanismus: Wenn ein Angreifer einen Wert manipuliert, ist das Scheitern der Authentifizierung für den FP eindeutig zu erkennen.

Auswirkungen: Die Authentifizierung scheitert und muss neu gestartet werden.

Gefahr: [auth-pp-jamming]	Verfahren zur Schadensbegrenzung

Schutzmechanismus: Da auf die AUTHENTICATION-REQUEST-Nachricht in jedem Fall eine Antwort erwartet wird, kann der FP erkennen, dass die Übertragung einer Nachricht gescheitert ist, wenn er keine Antwort erhält.

Auswirkungen: Die Authentifizierung scheitert und muss neu gestartet werden.

2.7.4 Authentifizierung eines FP

[EN 300 175-5, Abschnitt 13.3.3], [EN 300 175-7, Abschnitt 6.3.3.4]

Analog zur *Authentifizierung eines PP* kann ein PP auch die Authentifizierung eines FP durch ein Challenge-Response-Verfahren einfordern. Hierzu werden die Prozesse *A21* und *A22* des A Algorithm verwendet.

Initiiert wird diese Prozedur durch eine AUTHENTICATION–REQUEST-Nachricht, die ein PP zum FP sendet. In dieser Nachricht definiert der PP den zu verwendenden A Algorithm und einen Wert *RAND_P* der die vom FP zu lösende Challenge beinhaltet.

Wenn der FP diese Nachricht empfängt, prüft er zunächst, ob er die Anfrage bestätigt. Wenn beispielsweise der A Algorithm nicht unterstützt wird, lehnt er die Anfrage ab, indem er mit einer AUTHENTICATION–REJECT-Nachricht antwortet, in der eine Liste von alternativen A Algorithms enthalten ist.

Wenn die Anfrage bestätigt werden soll, lädt der FP zunächst den Wert von *RS* bzw. generiert einen neuen Wert für diesen Parameter und erzeugt seinerseits den zweiten Teil der Challenge: *RAND_F*. Nun lädt er – wenn er den *RS* wiederverwendet – den zuvor gespeicherten Wert von *KS*′ aus dem Speicher oder berechnet andernfalls den Wert von *KS* auf Basis des *A21*-Prozesses $KS' := \text{AES-128}_K(\text{AES-128}_K(RS))$.

Mit dem Prozess *A22* und den Werten *KS*′, *RAND_P* und *RAND_F* berechnet der FP anschließend die Response zur Challenge: $RES2 := \text{AES-128}_{KS}(RAND\_P \mid RAND\_F)$ Diese Response sendet er zusammen mit den weiteren von ihm festgelegten Parametern (*RS* und *RAND_F*) in der AUTHENTICATION–REPLY-Nachricht zum PP.

Der PP validiert diese Antwort, indem er seinerseits die *A21* und *A22* Prozesse mit allen Parametern berechnet. Die Authentifizierung ist erfolgreich, wenn die *expected Response XRES2* des FP mit der erhaltenen Response übereinstimmt.

Der Nachrichtenfluss einer erfolgreichen Durchführung dieses Szenarios ist in Abbildung 2.9 zu finden.

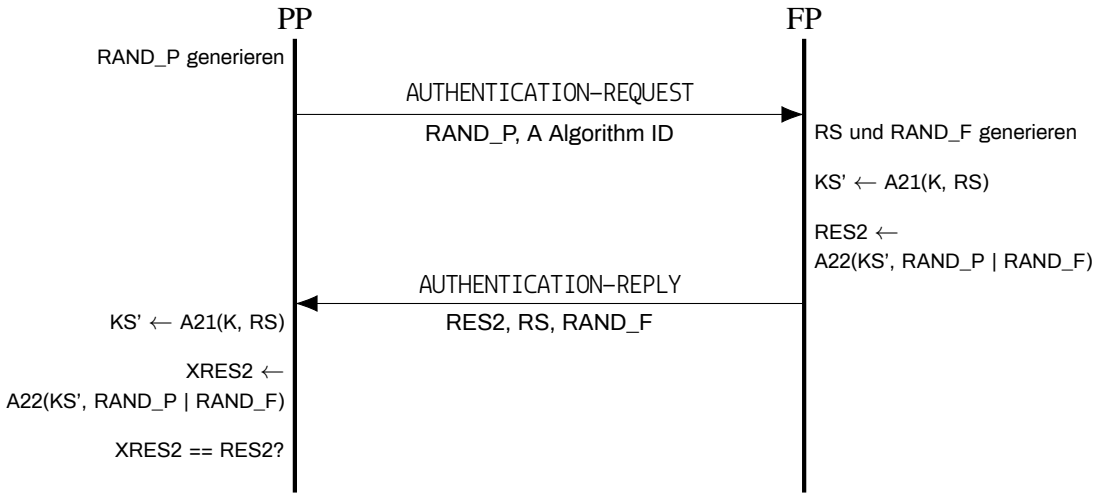


Abbildung 2.9: Nachrichtenfluss bei erfolgreicher *Authentifizierung eines FP*

Threat Model

Tabelle 2.11: Threat Model aus Sicht des PP für das *Authentifizierung eines FP*-Szenario

Sichtweise:	PP
Protokollszenario:	Authentifizierung eines FP
Spoofing:	• Ein Angreifer gibt sich als der zu authentifizierende FP aus und der PP authentifiziert daraufhin einen Angreifer. [auth-fp-spoofing-fp]
Tampering:	—
Information Disclosure:	• Ein Angreifer liest eine valide Response zu einer Challenge mit. [auth-fp-eavesdropping-challenge-response]
Denial of Service:	• Ein Angreifer manipuliert einen Wert, um die Authentifizierung scheitern zu lassen. [auth-fp-manipulating-value-dos] • Ein Angreifer stört die Kommunikation durch Jamming. [auth-fp-jamming]

Tabelle 2.12: Threat Model aus Sicht des FP für das *Authentifizierung eines FP*-Szenario

Sichtweise:	FP
Protokollszenario:	Authentifizierung eines FP
Spoofing:	—
Tampering:	• Ein Angreifer verändert den vom PP festgelegten A Algorithm. [auth-fp-downgrading-alg]
Information Disclosure:	• Ein Angreifer liest mit *RS*, *RAND_F* und *RAND_P* Teile des Schlüsselmaterials mit. [auth-fp-eavesdropping-key-material] • Ein Angreifer liest eine valide Response zu einer Challenge mit. [auth-fp-eavesdropping-challenge-response]
Denial of Service:	• Ein Angreifer manipuliert einen Wert, um die Authentifizierung scheitern zu lassen. [auth-fp-manipulating-value-dos] • Ein Angreifer stört die Kommunikation durch Jamming. [auth-fp-jamming]

Schutzmechanismen

Gefahr: [auth-fp-spoofing-fp] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Um sich als legitimer FP auszugeben und die Authentifizierung korrekt durchführen zu können, benötigt der Angreifer Kenntnis über den Wert des Authentication Key. Die Authentifizierung eines Angreifers scheitert, weil er diesen Wert nicht kennt.

Auswirkungen: Die Authentifizierung scheitert und muss neu initiiert werden.

Gefahr: [auth-fp-downgrading-alg] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Wenn ein Angreifer den A Algorithm in der `AUTHENTICATION-REQUEST`-Nachricht ändert, kann der PP diesen ablehnen. Dies tut er, indem er mit einer `AUTHENTICATION-REJECT`-Nachricht antwortet.

Wenn der FP den manipulierten A Algorithm verwendet, wird die Authentifizierung spätestens auf Seiten des PP scheitern, da er mit dem ursprünglich in der Nachricht kodierten A Algorithm einen anderen Wert für *RES2* berechnet, als der FP.

Ein schwacher A Algorithm (DSAA) sollte allerdings trotzdem zunächst abgewiesen werden. Gegebenenfalls kann ein Angreifer nämlich aus einem mit einem schwachen Algorithmus berechneten *RES2* Rückschlüsse auf den verwendeten Authentication Key ziehen.

Auswirkungen: Wenn der Angreifer den A Algorithm manipuliert, scheitert die Authentifizierung. Sie muss neu initiiert werden.

Gefahr: [auth-fp-eavesdropping-challenge-response]	verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Bei einem nicht gebrochenen A Algorithm kann ein Angreifer aus einer Response zu einer Challenge keine weiteren Informationen ableiten.

Gefahr: [auth-fp-eavesdropping-key-material]	verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Bei einem nicht gebrochenen A Algorithm kann ein Angreifer aus den Werten *RS*, *RAND_F* und *RAND_P* keine weiteren Informationen ableiten.

Gefahr: [auth-fp-manipulating-value-dos]	Verfahren zur Schadensbegrenzung

Schutzmechanismus: Das Scheitern der Authentifizierung ist für den PP eindeutig zu erkennen, allerdings kann der PP diese Situation nicht von der Situation eines gespooften FPs unterscheiden.

Auswirkungen: Das Szenario ist nicht erfolgreich und muss wiederholt werden. Möglicherweise bricht der PP die Kommunikation ab, weil er davon ausgeht, dass es sich um einen gefälschten FP handelt.

Gefahr: [auth-fp-jamming]	Verfahren zur Schadensbegrenzung

Schutzmechanismus: Da auf die AUTHENTICATION-REQUEST-Nachricht in jedem Fall eine Antwort erwartet wird, kann zumindest der PP erkennen, wenn die Nachricht verloren gegangen ist.

Auswirkungen: Das Szenario ist nicht erfolgreich und muss wiederholt werden.

2.7.5 Key Allocation

[EN 300 175-5, Abschnitt 13.6], [EN 300 175-7, Abschnitt 6.5.6]

Die *Key Allocation* ermöglicht die Etablierung eines neuen *UAK* auf Basis eines (schwächeren) *AC*. Dieser *AC* sollte laut Spezifikation mindestens eine Länge von 32 bit haben. Zur *Key Allocation* wird eine Variation beider Authentifizierungsprozeduren definiert, in der sich beide Kommunikationspartner gegenseitig authentifizieren.

Zu Beginn der Key Allocation wird der PP authentifiziert. Hierzu sendet der PP eine KEY-ALLOCATE-Nachricht an den PP. In dieser Nachricht legt der FP – wie bei dem Szenario *Authentifizierung eines PP* – den zu verwendenden A Algorithm fest und definiert die zu verwendenden Werte von *RS* und *RAND_F*. Hier unterscheidet sich nur der Nachrichtentyp KEY-ALLOCATE von dem im *Authentifizierung eines PP*-Protokollszenario verwendeten Nachrichtentyp AUTHENTICATION-REQUEST.

Wenn der PP diese Nachricht empfängt, prüft er, ob er mit dem gewählten A Algorithm einverstanden ist. Ist dies nicht der Fall, beendet er die Prozedur mit der Nachricht AUTHENTICATION-REJECT. Andernfalls

löst er die vom FP gestellte Challenge analog zur *Authentifizierung eines PP* in Abschnitt 2.7.3 „Authentifizierung eines PP“. Er generiert einen zufälligen Wert für *RAND_P* und berechnet dann mit den empfangenen Parametern *RS*, *RAND_F* und *RAND_P* die Prozesse *A11* und *A12*. Die Response zur Challenge *RAND_F* | *RAND_P* wird analog *RES1* genannt.

An dieser Stelle beginnt der PP mit dem Teil des Szenarios, das zur Authentifizierung des FP dient. Hierzu sendet er den von ihm generierten Teil der Challenge *RAND_P* in einer Nachricht vom Typ AUTHENTICATION–REQUEST an den FP. Gleichzeitig beweist der PP in dieser Nachricht seine Authentizität, indem er seinerseits die Antwort *RES1* auf die vom FP gestellte Challenge mit in dieser Nachricht verschickt. Der erste Teil der *Key Allocation* ist hiermit abgeschlossen und der PP ist authentifiziert. Zusätzlich legt der PP in dieser AUTHENTICATION–REQUEST-Nachricht denselben A Algorithm, den der FP zuvor definiert hat, fest.

Wenn der FP die AUTHENTICATION–REQUEST-Nachricht erhält, prüft er zunächst die empfangene Response *RES1*. Zur eigenen Berechnung dieses Wertes sind ihm alle benötigten Parameter bekannt: die zuvor von ihm gewählten Werte *RS* und *RAND_F* und der vom PP erzeugte Teil der Challenge *RAND_P*. Wenn der berechnete Wert für *RES1* nicht mit dem empfangenen Wert übereinstimmt, bricht der FP die *Key Allocation* ab. Andernfalls führt er den zweiten Teil des Szenarios durch: die *Authentifizierung eines FP*.

Um sich selbst gegenüber dem PP zu authentifizieren, muss der FP die vom PP gestellte Challenge *RAND_P* lösen. Hierzu generiert er – analog zu Abschnitt 2.7.4 „Authentifizierung eines FP“ – einen Wert für *RS* und einen Wert für *RAND_F* und berechnet mit diesem die Response zur Challenge *RAND_P*. In jedem Fall soll hier ein neuer Wert für *RS* generiert werden. Das Dokument *Part 5: Network (NWK) layer* [EN 300 175-5, Abschnitt 13.6] legt fest, dass auch ein neuer Teil der Challenge des FP, der *RAND_F*, erzeugt werden soll – das Dokument *Part 7: Security features* [EN 300 175-7, Abschnitt 6.5.6] macht hierzu keine Angaben.

Um die Response *RES2* zur Challenge *RAND_P* | *RAND_F* zu berechnen, muss der FP die Prozesse *A21* und *A22* ausführen:

$$KS' = \text{AES-128}_{AC}(\text{AES-128}_{AC}(RS)) \qquad (A21)$$

$$RES2 = \text{AES-128}_{KS'}(RAND\_P \mid RAND\_F) \qquad (A22)$$

Das Resultat KS' der Berechnung von *A11* speichert der FP nun als neuen *UAK*.

Zusammen mit der Response *RES2* schickt der FP die generierten Parameter *RS* und *RAND_F* in der AUTHENTICATION–REPLY-Nachricht an den PP, um seine Authentizität zu beweisen.

Bei Empfang dieser Nachricht prüft der PP die Response *RES2* des FP. Hierzu berechnet er analog den Wert für KS' und den Wert *RES2*. Wenn der vom PP berechnete Wert für *RES2* mit dem empfangenen Wert übereinstimmt, legt auch der PP den Wert KS' als neuen *UAK* fest.

In Abbildung 2.10 ist die Struktur dieses Szenarios mit den relevanten Berechnungen grafisch aufbereitet.

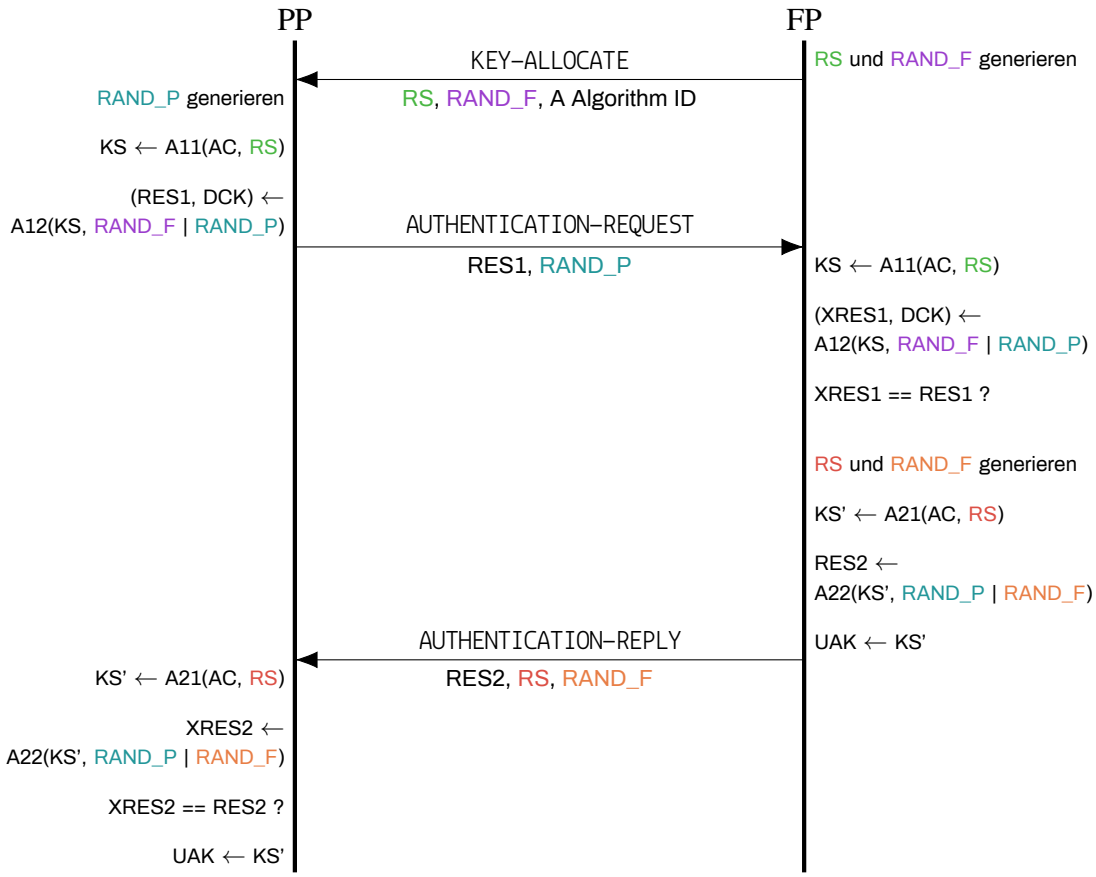


Abbildung 2.10: Nachrichtenfluss bei erfolgreicher *Key Allocation*

Threat Model

Tabelle 2.13: Threat Model aus Sicht des PP für das *Key Allocation*-Szenario

Sichtweise:	PP
Protokollszenario:	Key Allocation
Spoofing:	• Ein Angreifer gibt sich als FP aus, sodass der PP die *Key Allocation* mit dem Angreifer durchführt, anstelle mit der validen Basisstation. [key-allocation-spoofing-fp]
Tampering:	• Ein Angreifer manipuliert den zu verwendenden A Algorithm. [key-allocation-downgrading]
Information Disclosure:	• Ein Angreifer liest zwei valide Responses zu zwei Challenges mit. [key-allocation-eavesdropping-challenge-response]
Denial of Service:	• Ein Angreifer sendet eine AUTHENTICATION_REJECT-Nachricht, um das Szenario abzubrechen. [key-allocation-injecting-authentication-reject] • Ein Angreifer manipuliert einen Wert, um die Authentifizierung scheitern zu lassen. [key-allocation-value-manipulation-dos] • Ein Angreifer stört die Kommunikation durch Jamming. [key-allocation-jamming]

Tabelle 2.14: Threat Model aus Sicht des FP für das *Key Allocation*-Szenario

Sichtweise:	FP
Protokollszenario:	Key Allocation
Spoofing:	• Ein Angreifer gibt sich als PP aus, sodass der FP die *Key Allocation* mit dem Angreifer durchführt, anstelle mit dem PP. [key-allocation-spoofing-pp] • Ein Angreifer fügt eine AUTHENTICATION_REJECT-Nachricht ein, um die Verwendung eines anderen A Algorithm zu erzwingen. [key-allocation-injecting-authentication-reject-a-alg]
Tampering:	—
Information Disclosure:	• Ein Angreifer liest zwei valide Responses zu zwei Challenges mit. [key-allocation-eavesdropping-challenge-response]
Denial of Service:	• Ein Angreifer manipuliert einen Wert, um die Authentifizierung scheitern zu lassen. [key-allocation-value-manipulation-dos] • Ein Angreifer stört die Kommunikation durch Jamming. [key-allocation-jamming]

Schutzmechanismen

Gefahr: [key-allocation-spoofing-fp] | optionales Verfahren zur Abwehr

Schutzmechanismus: Ein Angreifer kann sich während der *Key Allocation* nicht als FP ausgeben, da er hierzu den verwendeten *AC* bzw. Authentication Key kennen müsste.

Das Security-Dokument der DECT Spezifikation sieht vor, dass die *Key Allocation* mit einem *AC* durchgeführt wird, der eine Mindestlänge von 32 bit hat („the AC from which the UAK is derived should be at least 32 bits long“, siehe [EN 300 175-7, Abschnitt 6.5.6.1]). Der *AC* ist allerdings als „short value“ designed, sodass kurze Werte üblich sind. So gibt dasselbe Dokument an einer anderen Stelle an, dass der AC üblicherweise aus 16 bis 32 bit besteht („This is usually a short value (for example 16 bits to 32 bits)“, siehe [EN 300 175-7, Abschnitt 4.4.3.1]).

Dadurch, dass ein Angreifer diesen Wert allerdings nicht kennt, ist er nicht in der Lage, die Key Allocation stellvertretend für den FP durchzuführen, da sie aller Wahrscheinlichkeit nach scheitern würde. Die Wahrscheinlichkeit, dass ein Angreifer den korrekten Wert für den *AC* errät, ist extrem gering und liegt bei $\frac{1}{2^{32}}$. Im Mittel müsste die *Key Allocation* also 2^{31} mal durchgeführt werden, damit ein Angreifer den FP erfolgreich fälschen kann. Da ein Angreifer in der Position des FP diese Prozedur jedoch initiieren kann, ist die Gefahr eines Brute-Force Angriffs hier nicht gebannt.

Wenn der PP die *Key Allocation* jedoch nur als Nested Procedure während des *Obtain Access Rights*-Szenarios zulässt, kann ein Angreifer das Szenario nicht beliebig häufig starten, um beliebig viele Versuche zum Raten des AC zu erhalten. Wenn die *Key Allocation* als Nested Procedure fehlschlägt, schlägt auch das *Obtain Access Rights*-Szenario fehl.

Auswirkungen: Die *Key Allocation* scheitert und muss neu gestartet werden.

Gefahr: [key-allocation-spoofing-pp] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Diese Gefahr ist sehr ähnlich zu der Gefahr [key-allocation-spoofing-fp]. Aus Sicht des FP ist ein Angreifer hier allerdings nicht in der Lage, die *Key Allocation* zu initiieren. Auf diese Weise kann er bei Scheitern des Szenarios das Szenario nicht einfach erneut starten, um einen neuen Versuch beim Erraten des *AC* zu erhalten.

Auswirkungen: Die *Key Allocation* scheitert und muss neu gestartet werden.

Gefahr: [key-allocation-injecting-authentication-reject-a-alg] | vorgeschlagene Maßnahme

Schutzmechanismus: Ein FP kann die *Key Allocation* erneut initiieren und dabei den in der `AUTHENTICATION-REJECT`-Nachricht empfangenen A Algorithm ignorieren. Im Zweifel verhindert dieses Verhalten allerdings die Kompatibilität des FP zu PPs die ausschließlich den anderen A Algorithm verwenden.

Auswirkungen: Die *Key Allocation* scheitert und muss neu gestartet werden.

Gefahr: [key-allocation-downgrading] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Wenn ein Angreifer den zu verwendenden A Algorithm ändert, kann dieser vom Kommunikationspartner abgelehnt werden. Dies tut er, indem er mit einer AUTHENTICATION_REJECT-Nachricht antwortet.

Wenn der vom Angreifer eingeschleuste A Algorithm verwendet wird, wird die Authentifizierung spätestens bei der Überprüfung der Response scheitern, da sie mit einem anderen A Algorithm überprüft werden würde, als sie berechnet wurde.

Ein schwacher A Algorithm (DSAA) sollte allerdings trotzdem zunächst abgewiesen werden. Gegebenenfalls kann ein Angreifer nämlich aus einer mit einem schwachen Algorithmus berechneten Response Rückschlüsse auf den verwendeten Authentication Key oder KS' ziehen.

Auswirkungen: Wenn der Angreifer den A Algorithm manipuliert, scheitert die Authentifizierung. Sie muss neu initiiert werden.

Gefahr: [key-allocation-eavesdropping-challenge-response] | vorgeschlagene Maßnahme

Schutzmechanismus: Aus zwei validen Responses und zwei Challenges lässt sich der Authentication Key mithilfe von Brute-Force erraten. Da der in diesem Szenario verwendete Authentication Key in der Regel eine Länge von 32 bit hat, ist es für einen Angreifer möglich, auf Basis der Challenge (*RAND_P*, *RAND_F* und *RS*) und der dazugehörigen Response (*RES1* bzw. *RES2*) nachträglich den verwendeten *AC* zu bestimmen. Hierzu sind im Mittel lediglich 2^{31} Versuche notwendig, was die Durchführung eines solchen Angriffs praktikabel macht.

Wenn die *Key Allocation* jedoch mit einem stärkeren Schlüssel durchgeführt werden würde, würde ein Brute-Force Angriff zur Berechnung des Schlüssels nicht in vertretbarer Zeit gelingen. Grundsätzlich lässt die Spezifikation den Austausch des *AC* auf einem beliebigen Weg zu, weshalb beispielsweise ein Out-of-Band Schlüsselaustausch, bspw. auf Basis von NFC oder QR-Codes vor der *Key Allocation* dürchgeführt werden könnte. Wenn dieser *AC* eine Länge von bspw. 128 bit hätte, wäre davon auszugehen, dass ein Angreifer diesen Wert nicht in vertretbarer Zeit mit Brute-Force Methoden erraten kann.

Auswirkungen: Ein Angreifer kann sich sowohl dem PP gegenüber als FP ausgeben, als auch dem FP gegenüber als PP, wenn es ihm gelingt, den korrekten Authentication Key zu erraten. Des Weiteren ist er in der Lage, jede Kommunikation zwischen dem FP und dem PP zu entschlüsseln.

Gefahr: [key-allocation-injecting-authentication-reject] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Die *Key Allocation* scheitert, wenn einer der Partner eine AUTHENTICATION_REJECT-Nachricht im Namen der Gegenseite empfängt. Da in diesem Fall allerdings auch der *UAK* nicht auf einen falschen Wert aktualisiert wird, ist die Authentifizierung der Kommunikationsteilnehmer nicht nachhaltig beeinträchtigt, da die Kommunikationspartner den alten (ggf. allerdings schwächeren) Authentication Key weiter verwenden können.

Auswirkungen: Die *Key Allocation* scheitert und muss neu gestartet werden.

Gefahr: [key-allocation-value-manipulation-dos] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Wenn Werte während der Übertragung manipuliert werden, wird die Prüfung einer Response fehlschlagen. An diesem Punkt scheitert die *Key Allocation*. Wenn die Prüfung von *RES1* fehlschlägt, bemerkt der FP das Scheitern des Szenarios und versendet die AUTHENTICATION–REJECT-Nachricht, um den PP über das Scheitern zu informieren.

Wenn die Prüfung von *RES2* fehlschlägt, bemerkt der PP das Scheitern und aktualisiert den Wert für den *UAK* nicht. Im Nachhinein wird dann keine Authentifizierung mit dem neuen *UAK* zustande kommen. Beide Kommunikationspartner würden in diesem Fall den alten (ggf. allerdings schwächeren) Authentication Key weiter verwenden und ggf. eine neue *Key Allocation* einleiten.

Auswirkungen: Die *Key Allocation* scheitert und muss neu gestartet werden.

Gefahr: [key-allocation-jamming] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Jamming kann durch beide Kommunikationspartner erkannt werden.

a) Wenn die KEY–ALLOCATE-Nachricht gestört wird, kann der FP dies erkennen, da der PP nicht mit einer AUTHENTICATION–REQUEST-Nachricht antwortet.

b) Wenn die AUTHENTICATION–REQUEST-Nachricht gestört wird, kann der PP dies erkennen, da der FP nicht mit einer AUTHENTICATION–REPLY-Nachricht oder einer AUTHENTICATION–REJECT-Nachricht antwortet.

c) Wenn die AUTHENTICATION–REPLY-Nachricht gestört wird, kann dies zunächst nicht erkannt werden. Der PP wird in diesem Fall allerdings den *UAK* nicht berechnen, sodass die *Key Allocation* auf Seiten des PP nie abgeschlossen wird.

   Die *Key Allocation* wird allerdings in letzter Konsequenz nur als erfolgreich abgeschlossen gewertet, wenn die nächste Authentifizierung eines Partners mithilfe des neuen *UAK* problemlos verläuft. Wenn diese nächste Authentifizierung fehlschlägt, gehen die Kommunikationspartner davon aus, dass die *Key Allocation* nicht erfolgreich war und fallen für die nachfolgende Kommunikation auf die Nutzung des alten Authentication Key zurück.

Aus diesem Grund ist nicht die gesamte nachfoglende Authentifizierung unmöglich, da in jedem Fall beide Kommunikationspartner den alten (ggf. allerdings schwächeren) Authentication Key weiter verwenden und ggf. eine neue *Key Allocation* einleiten.

Auswirkungen: Die *Key Allocation* scheitert und muss neu gestartet werden.

2.7.6 Easy Pairing

[TS 102 939-1, Abschnitt 14.1]

Für ULE ist ein besonderes Szenario, in dem die *Obtain Access Rights Procedure* mit der *Key Allocation* verzahnt ist, definiert. *Easy Pairing* ermöglicht es, das Pairing eines PP mit einem FP durchzuführen, ohne vorher manuell einen *AC* auszutauschen, indem es als *AC* den festen Wert 0000 nutzt. Wie beim *Obtain Access Rights*-Szenario muss der FP zunächst den Beitritt von neuen Netzwerkteilnehmern erlauben und das *Access Rights Supported*-Bit in dem System Information Broadcast setzen.

Wenn ein PP nun dem Netz beitreten möchte, startet – sofern vom FP unterstützt – es das *Obtain Access Rights*-Szenario, wie in Abschnitt 2.7.1 „Obtaining Access Rights“ definiert. Beim *Easy Pairing* wird hier die Ausführung der *Key Allocation* als Nested Procedure allerdings nicht optional definiert, sondern verpflichtend. Zudem wird, abweichend zu Abschnitt 2.7.5 „Key Allocation“, nicht gefordert, dass der *AC* mindestens 32 bit lang ist. Dem *AC* wird der feste Wert 0000 zugewiesen. Mit diesem Szenario soll es für jeden PP möglich sein, einen starken *UAK* zu etablieren, auch wenn einfache PPs, wie beispielsweise Fenstersensoren, keine Eingabemöglichkeit für Out-of-Band *AC*s besitzen.

Der Nachrichtenfluss einer erfolgreichen Durchführung des Easy Parirings dieses Szenarios ist in Abbildung 2.11 zu finden; Teile in kursiv sind optional.

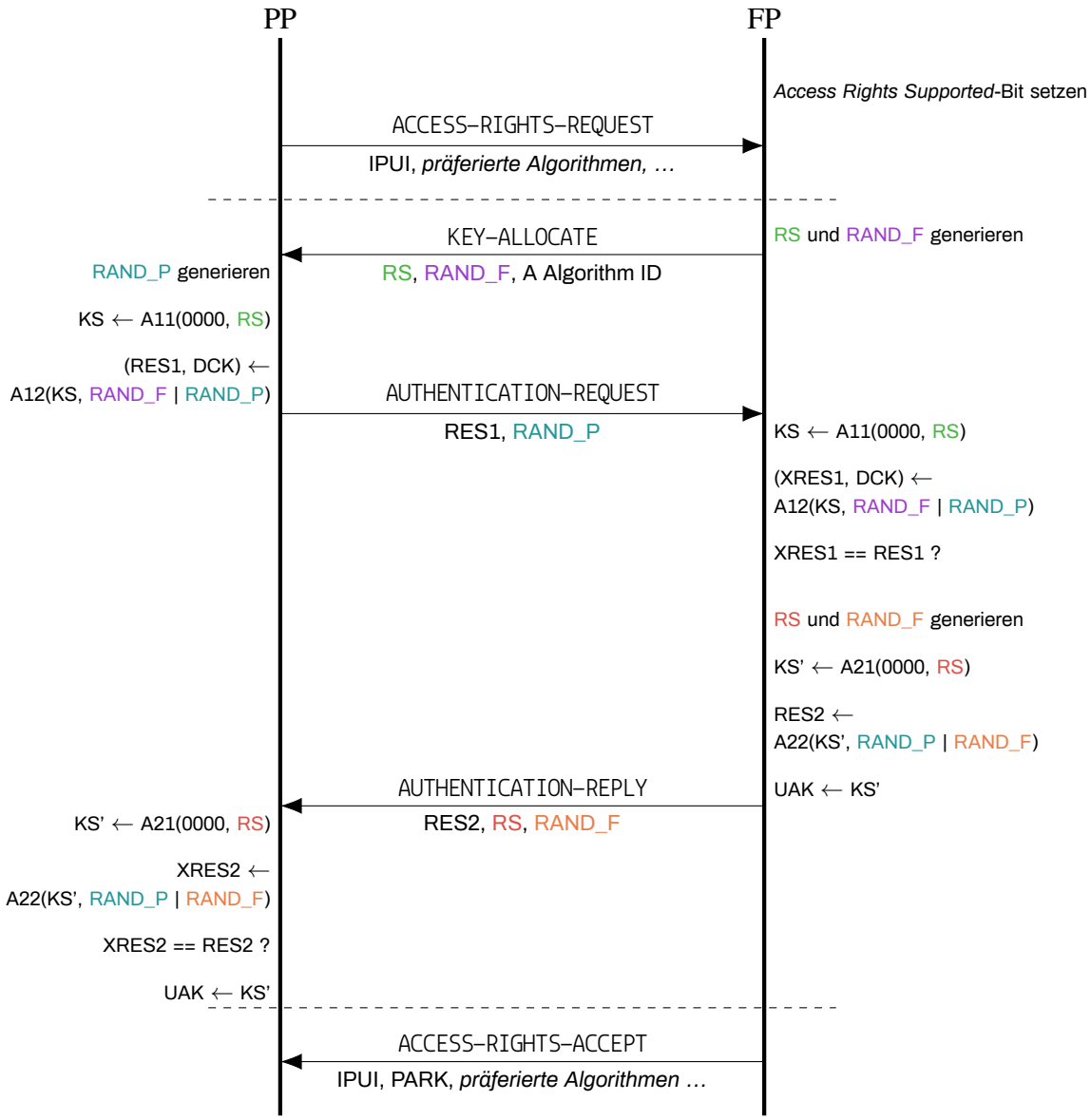


Abbildung 2.11: Nachrichtenfluss bei erfolgreichem *Easy Pairing*

Threat Model

Tabelle 2.15: Threat Model aus Sicht des PP für das *Easy Pairing*-Szenario

Sichtweise:	PP
Protokollszenario:	Easy Pairing
Spoofing:	• Ein Angreifer gibt sich als FP aus. Der PP verbindet sich daraufhin mit einem von einem Angreifer aufgespannten Netzwerk. [easy-pairing-spoofing-fp] • Ein Angreifer führt einen Man-in-the-Middle Angriff durch. Er gibt sich gleichzeitig dem PP gegenüber als FP aus und dem FP gegenüber als PP. Daraufhin verbindet sich der legitime PP nicht mit dem FP, sondern mit dem Angreifer. [easy-pairing-mitm]
Tampering:	• Ein Angreifer manipuliert den zu verwendenden A Algorithm. [easy-pairing-downgrading]
Information Disclosure:	• Ein Angreifer berechnet dieselben Algorithmen, die auch der PP und FP nutzen und berechnet so den geheimen abgeleiteten *UAK*. [easy-pairing-eavesdropping-keys]
Denial of Service:	• Ein Angreifer sendet eine AUTHENTICATION–REJECT-Nachricht an einen Kommunikationsteilnehmer. [easy-pairing-injecting-authentication-reject] • Ein Angreifer sendet eine ACCESS–RIGHTS–REJECT-Nachricht an einen Kommunikationsteilnehmer. [easy-pairing-injecting-access-rights-reject] • Ein Angreifer manipuliert einen Wert (*RAND_F*, *RAND_P*, *RS*, *RES1*, *RES2* oder den zu verwendenden A Algorithm). [easy-pairing-manipulating-value-dos] • Ein Angreifer stört die Kommunikation durch Jamming. [easy-pairing-jamming]

Tabelle 2.16: Threat Model aus Sicht des FP für das *Easy Pairing*-Szenario

Sichtweise:	FP
Protokollszenario:	Easy Pairing
Spoofing:	• Ein Angreifer gibt sich als PP aus und tritt dem Netzwerk bei. [easy-pairing-spoofing-pp] • Ein Angreifer führt einen Man-in-the-Middle Angriff durch. Er gibt sich gleichzeitig dem PP gegenüber als FP aus und dem FP gegenüber als PP. Daraufhin verbindet sich der legitime PP nicht mit dem FP, sondern mit dem Angreifer. [easy-pairing-mitm]
Tampering:	—
Information Disclosure:	• Ein Angreifer berechnet dieselben Algorithmen, die auch der PP und FP nutzen und berechnet so den geheimen abgeleiteten *UAK*. [easy-pairing-eavesdropping-keys]
Denial of Service:	• Ein Angreifer sendet eine AUTHENTICATION_REJECT-Nachricht an einen Kommunikationsteilnehmer. [easy-pairing-injecting-authentication-reject] • Ein Angreifer manipuliert einen Wert (*RAND_F*, *RAND_P*, *RS*, *RES1*, *RES2* oder den zu verwendenden A Algorithm). [easy-pairing-manipulating-value-dos] • Ein Angreifer manipuliert die IPUI des PP. [easy-pairing-manipulating-ipui] • Ein Angreifer stört die Kommunikation durch Jamming. [easy-pairing-jamming]

Schutzmechanismen

Gefahr: [easy-pairing-spoofing-fp] | kein Schutzmechanismus

Bemerkung: Ein ULE Gerät ist während des *Easy Pairing* nicht vor gefälschten FPs sicher.

Auswirkungen: Wenn ein Angreifer sich während des *Easy Pairing* als FP ausgibt, tritt der PP einem von einem Angreifer betriebenen Netzwerk bei. Der Schlüsselaustausch verläuft fehlerfrei und der PP ist nicht in der Lage zu erkennen, dass es sich bei dem Angreifer nicht um den korrekten FP handelt. Der User ist als einziger in der Lage eine Fehlfunktion des Gerätes festzustellen, da keine Kommunikation mit dem ordnungsgemäßen FP stattfindet.

Gefahr: [easy-pairing-mitm] | kein Schutzmechanismus

Bemerkung: Bei einem Man-in-the-Middle-Angriff ist ein Opfer (PP bzw. FP) nicht in der Lage zu bemerken, dass die Verbindung unsicher ist, da – sofern der Angreifer seinen Angriff fortsetzt – die nachfolgenden Authentifizierungsverfahren korrekt ablaufen und auch die Befehle korrekt weitergeleitet und durchgeführt werden.

Auswirkungen: Ein Angreifer hat nach dem in Abschnitt 2.3 „Bedrohungsmodell“ definierten Modell die volle Kontrolle über die Kommunikation des PP/FP-Paars. Wenn er die Nachrichten nicht weiterleitet, sind PP und FP nicht in der Lage sich gegenseitig zu authentifizieren, sodass keine direkte Kommunikation zwischen den legitimen Partnern stattfinden kann.

Gefahr: [easy-pairing-spoofing-pp] | kein Schutzmechanismus

Bemerkung: Da keine weitere Authentifizierung stattfindet, wenn der FP das *Easy Pairing* anbietet, kann jedes Gerät ohne Prüfung dem Netz beitreten.

Auswirkungen: Ein Angreifer hat nach dem im Abschnitt 2.3 „Bedrohungsmodell“ definierten Modell volle Kontrolle über das gesamte Netz. Er hat jedoch keine Kenntnis über die Werte der Authentication Keys der anderen PPs und kann somit keine Kommunikation manipulieren.

Gefahr: [easy-pairing-downgrading] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Wenn ein Angreifer den A Algorithm in der `AUTHENTICATION-REQUEST`-Nachricht ändert, kann der PP diesen ablehnen. Dies tut er, indem er mit einer `AUTHENTICATION-REJECT`-Nachricht antwortet.

Wenn der FP den manipulierten A Algorithm verwendet, wird die Authentifizierung spätestens auf Seiten des PP scheitern, da er mit dem ursprünglich in der Nachricht kodierten A Algorithm einen anderen Wert für *RES2* berechnet, als der FP.

Auswirkungen: Wenn der Angreifer den A Algorithm manipuliert, scheitert die Authentifizierung. Sie muss neu initiiert werden.

Gefahr: [easy-pairing-eavesdropping-keys] | kein Schutzmechanismus

Bemerkung: Beim *Easy Pairing* wird explizit kein Out-of-Band ausgetauschter *AC* verwendet, um den Schlüsselaustausch abzusichern, sondern stattdessen der statische Wert 0000. Auf diese Weise kann ein Angreifer problemlos dasselbe Schlüsselmaterial ableiten, das die beiden Kommunikationspartner berechnen.

Auswirkungen: Der Angreifer ist in der Lage, in der nachfolgenden Kommunikation sowohl den PP als auch den FP zu spoofen. Wenn der Angreifer das *Authentifizierung eines PP*-Szenario einer Kommunikation abhört, ist er auch in der Lage die Kommunikation der nachfolgenden Kommunikation mitzulesen, zu entschlüsseln und zu manipulieren, da er aus den dort mitgelesenen Werten auch den *DCK* einer Verbindung berechnen kann.

Gefahr: [easy-pairing-injecting-authentication-reject] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Während des *Easy Pairing* Szenario kann ein Angreifer Nachrichten in die Kommunikation injizieren, da diese weder integritätsgesichert noch verschlüsselt sind.

Wenn ein Kommunikationsteilnehmer eine AUTHENTICATION–REJECT-Nachricht empfängt, scheitert das *Easy Pairing*. Danach kann das *Easy Pairing*-Szenario wieder gestartet werden. Der Empfänger der AUTHENTICATION–REJECT registriert das Scheitern des Szenarios.

Auswirkungen: Das *Easy Pairing* scheitert und muss neu gestartet werden.

Gefahr: [easy-pairing-injecting-access-rights-reject] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Während des *Easy Pairing* Szenario kann ein Angreifer Nachrichten in die Kommunikation injizieren, da diese weder integritätsgesichert noch verschlüsselt sind.

Wenn ein Kommunikationsteilnehmer eine ACCESS–RIGHTS–REJECT-Nachricht empfängt, scheitert das *Easy Pairing*. Danach kann das *Easy Pairing*-Szenario wieder gestartet werden. Der Empfänger der ACCESS–RIGHTS–REJECT registriert das Scheitern des Szenarios.

Auswirkungen: Das *Easy Pairing* scheitert und muss neu gestartet werden.

Gefahr: [easy-pairing-manipulating-value-dos] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Wenn Werte wie die Challenge verändert werden, berechnen beide Kommunikationspartner verschiedene Werte für die Response zur Challenge. Der Sender berechnet die Response mit dem nicht-manipulierten Wert und der Empfänger nutzt den manipulierten Wert dazu. Da die Responses dann nicht identisch sind, scheitert das *Easy Pairing*.

Auswirkungen: Wenn das übertragene Schlüsselmaterial manipuliert wird, scheitert das *Easy Pairing*, weil beide Partner unterschiedliche Werte von *RES1* bzw. *RES2* berechnen.

Gefahr: [easy-pairing-jamming] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Auf alle Nachrichten, bis auf die Nachrichten AUTHENTICATION–REPLY UND ACCESS–RIGHTS–ACCEPT vom FP werden Antworten erwartet. Damit ist es den Kommunikationsteilnehmern zumindest möglich, herauszufinden, dass Nachrichten verloren gegangen sind, wenn keine Antwort ankommt.

Auswirkungen: Das *Easy Pairing* muss neu gestartet werden.

Gefahr: [easy-pairing-manipulating-ipui]	kein Schutzmechanismus

Bemerkung: Während des *Easy Pairing* Szenario kann ein Angreifer Nachrichten gezielt manipulieren, da diese weder integritätsgesichert noch verschlüsselt sind.

Auswirkungen: Der PP kann sich nach dem Easy Pairing nicht mit dem FP verbinden, da dieser keine Senderechte für die angegebene IPUI gespeichert hat.

2.7.7 Starten der MAC Encryption

[EN 300 175-7, Abschnitt 6.4.6.3]

DECT unterstützt die Verschlüsselung aller Nutzdaten auf dem MAC-Layer. Um mit dieser Verschlüsselung zu beginnen, muss zunächst ein Schlüssel ausgetauscht werden. DECT definiert hier zwei verschiedene Vorgehensweisen: den Aufbau einer Verbindung mit der sogenannten *Early Encryption* oder den Verbindungsaufbau im Klartext, die nachträglich verschlüsselt werden kann.

1. Wenn bereits bei einer vorherigen Verbindung ein Schlüssel ausgetauscht worden ist, der explizit als *DefCK* (Default Cipher Key) markiert wurde, kann der MAC-Layer diesen Schlüssel direkt nachdem die MAC-Verbindung aufgebaut wurde dazu nutzen, um das *Starten der MAC Encryption*-Szenario zu starten und damit die Verbindung zu verschlüsseln. Sobald die Verschlüsselung aktiv ist, sollte jedoch das Szenario *Authentifizierung eines PP* ausgeführt werden, um einen neuen kurzlebigen Session-Schlüssel (*DCK*) zu erzeugen.
2. Wenn die *Early Encryption* nicht vorbereitet wurde, muss die Verbindung im Klartext aufgebaut werden. In diesem Fall ist entweder der erste Payload einer MAC-Connection so lange unverschlüsselt, bis das *Authentifizierung eines PP*-Szneario durchgeführt wurde, um einen *DCK* abzuleiten und danach das *Starten der MAC Encryption*-Szenario durchgeführt wurde.

Um die Verschlüsselung einer MAC-Verbindung zu aktivieren, sendet der PP die MAC-Layer Nachricht `START.REQ`[2] im A-Field eines regulären Pakets innerhalb eines Frames mit gerader Framenummer an den FP. Das B-Field dieser Nachricht trägt dabei weiterhin regulären Payload. Der FP antwortet darauf mit der `START.CONF`[3]-Nachricht, die ebenso nur im A-Field eines Pakets übertragen wird. Diese Antwort muss in einem Frame mit ungerader Framenummer stattfinden. Alle Nutzdaten, die nach dieser Nachricht vom FP versendet werden, werden verschlüsselt – auch der Payload des B-Fields der Nachricht, die die `START.CONF`-Nachricht im A-Field trägt. Wenn der PP diese Nachricht empfängt, antwortet er im darauffolgenden Frame (bzw. nächsten Frame mit gerader Framenummer) mit der `START.GRANT`[4]-Nachricht. Ab dieser Nachricht verschlüsselt der PP auch seinen Payload und die Verschlüsselung der MAC-Verbindung ist aufgebaut.

Wenn dieses Verfahren zur Initiierung der Early Encryption genutzt wird, kann der FP die Verschlüsselung ablehnen, indem er die Nachricht `START.REJECT`[5] sendet.

[2] In [EN 300 175-3] wird die Nachricht `start encryption with cipher key-index : request` genannt
[3] In [EN 300 175-3] wird die Nachricht `start encryption with cipher key-index : confirm` genannt
[4] In [EN 300 175-3] wird die Nachricht `start encryption with cipher key-index : grant` genannt
[5] In [EN 300 175-3] wird die Nachricht `start encryption with cipher key-index : reject` genannt

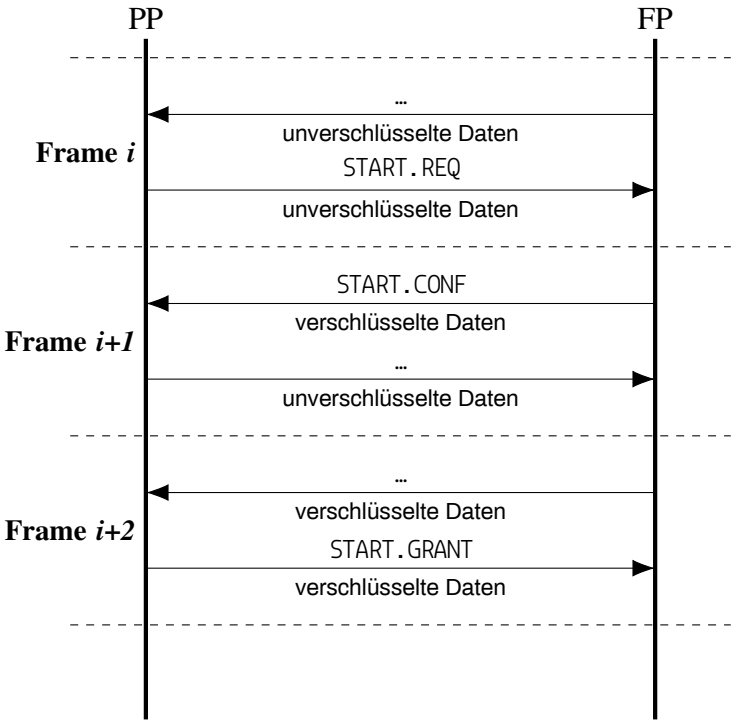


Abbildung 2.12: Idealer Nachrichtenfluss bei erfolgreichem Start der MAC-Verschlüsselung

Der vollständige Nachrichtenfluss bei fehlerfreier Übertragung ist in Abbildung 2.12 dargestellt.

Threat Model

Tabelle 2.17: Threat Model aus Sicht des PP für das *Start MAC Encryption*-Szenario

Sichtweise:	PP
Protokollszenario:	Start MAC Encryption
Spoofing:	—
Tampering:	• Ein Angreifer manipuliert die übertragenen Daten. [start-mac-encryption-manipulating-data]
Information Disclosure:	• Ein Angreifer liest unverschlüsselte Daten mit. [start-mac-encryption-eavesdropping-data]
Denial of Service:	• Ein Angreifer stört die Kommunikation durch Jamming. [start-mac-encryption-jamming]

Tabelle 2.18: Threat Model aus Sicht des FP für das *Start MAC Encryption*-Szenario

Sichtweise:	FP
Protokollszenario:	Start MAC Encryption
Spoofing:	—
Tampering:	• Ein Angreifer manipuliert die übertragenen Daten. [start-mac-encryption-manipulating-data]
Information Disclosure:	• Ein Angreifer liest unverschlüsselte Daten mit. [start-mac-encryption-eavesdropping-data]
Denial of Service:	• Ein Angreifer stört die Kommunikation durch Jamming. [start-mac-encryption-jamming] • Ein Angreifer injiziert eine `START.REJECT` Nachricht. [start-mac-encryption-injecting-start-reject] • Ein Angreifer kann die `START.REQ`-Nachricht so manipulieren, dass sie beim FP nicht als `START.REQ`-Nachricht ankommt. In diesem Fall kommt die Verschlüsselung nicht zustande. [start-mac-encryption-jamming-start-req]

Schutzmechanismen

Gefahr: [start-mac-encryption-manipulating-data] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Die MAC Encryption schützt grundsätzlich nicht vor Datenmanipulation. Die Daten werden nicht signiert und könnten unbemerkt manipuliert werden.

Bei ULE werden Applikationsdaten jedoch zunächst mit AES-CCM verschlüsselt. Das ermöglicht die Erkennung von Datenmanipulationen im DLC Layer. Manipulierte Daten werden nicht weiter verarbeitet.

Auswirkungen: Der Empfänger der Daten erkennt die Manipulation und verwirft die Daten, ohne deren Empfang zu bestätigen.

Gefahr: [start-mac-encryption-eavesdropping-data] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Die Spezifikation sieht vor, dass, während die Verschlüsselung ausgehandelt wird, weiterhin Nachrichten versendet werden können. Die Anfragen zum Verschlüsselungsstart werden vollständig im A-Field übermittelt, sodass im B-Field zu diesem Zeitpunkt weiterhin unverschlüsselt Nutzdaten übertragen werden können.

Bei ULE werden Applikationsdaten jedoch unabhängig von der MAC-Encryption bereits auf dem DLC-Layer mit AES-CCM verschlüsselt. Die hier übertragenen Daten sind also bereits verschlüsselt und können von einem Angreifer nicht mitgelesen werden, ohne dass dieser den genutzten Schlüssel kennt.

Darüber hinaus bestünde für DECT-Geräte, die keine AES-CCM Verschlüsselung nutzen, die Möglichkeit, die B-Fields der Kommunikation mit *Null-Messages* zu füllen, so lange die Verschlüsselung

noch nicht vollständig aufgebaut ist. Aus diesen Nachrichten könnte ein Angreifer keine Informationen gewinnen.

Gefahr: [start-mac-encryption-jamming] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Die MAC-Verschlüsselung kommt nicht zustande. Dies ist mindestens für die initiierende Partei (den PP) zu erkennen, da die `START.CONF`-Nachricht nicht empfangen wird. Der PP kann die Prozedur erneut initiieren.

Wenn DECT Kommunikation nur auf MAC-Verschlüsselung setzt, kann die Datenübertragung so lange pausiert werden, bis die Verschlüsselung aktiv ist, um keine Informationen preiszugeben. Hierzu können Applikationen beispielsweise die B-Fields der Nachrichten mit *Null-Messages* füllen.

Auswirkungen: Die MAC-Verschlüsselung wird nicht gestartet. Die AES-CCM Verschlüsselung der Applikationsdaten auf dem DLC-Layer ist davon unabhängig.

Gefahr: [start-mac-encryption-injecting-start-reject] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Bei ULE werden Applikationsdaten unabhängig von der MAC-Encryption mit AES-CCM verschlüsselt. Diese Verschlüsselung kommt unabhängig von diesem Szenario zustande.

Auswirkungen: Die MAC-Verschlüsselung wird nicht gestartet. Die AES-CCM Verschlüsselung der Applikationsdaten auf dem DLC-Layer ist davon unabhängig.

Gefahr: [start-mac-encryption-jamming-start-req] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Bei ULE werden Applikationsdaten unabhängig von der MAC-Encryption mit AES-CCM verschlüsselt. Diese Verschlüsselung kommt unabhängig von diesem Szenario zustande.

Auswirkungen: Die MAC-Verschlüsselung wird nicht gestartet. Die AES-CCM Verschlüsselung der Applikationsdaten auf dem DLC-Layer ist davon unabhängig.

2.7.8 Terminieren der MAC Encryption

[EN 300 175-7, Abschnitt 6.4.6.4]

Die MAC-Verschlüsselung einer Verbindung kann mithilfe des *Terminieren der MAC Encryption*-Szenarios beendet werden. Nach erfolgreicher Durchführung dieser Prozedur versenden beide Kommunikationsteilnehmer ihre Nachrichten im Klartext.

Das Terminieren einer MAC-Encryption ist ähnlich zu der Initiierung der Verschlüsselung. Zunächst sendet der PP eine `STOP.REQ`-Nachricht in einem Frame mit gerader Framenummer an den FP. Wenn dieser die Nachricht empfängt, antwortet der FP in einem Frame mit ungerader Framenummer mit der `STOP.CONF`-Nachricht. Mit dem Versand der `STOP.CONF`-Nachricht beginnt der FP damit den folgenden Payload im

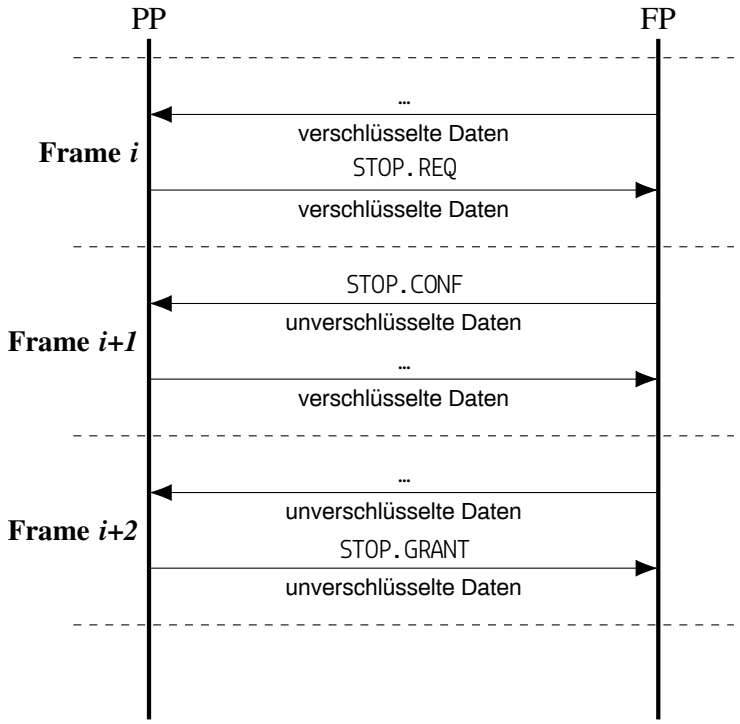


Abbildung 2.13: Idealer Nachrichtenfluss bei erfolgreicher Terminierung der MAC-Verschlüsselung

Klartext zu versenden. Hierzu zählt auch das B-Field der Nachricht, in der die STOP.CONF-Nachricht versendet wird.

Im nächsten Frame nach dem Erhalt der STOP.CONF-Nachricht, das eine gerade Framenummer hat, antwortet der PP mit der STOP.GRANT-Nachricht. Alle B-Fields, die der STOP.CONF-Nachricht folgen, werden vom PP im Klartext übertragen.

Der vollständige Nachrichtenfluss bei fehlerfreier Übertragung ist in Abbildung 2.13 dargestellt.

Threat Model

Tabelle 2.19: Threat Model aus Sicht des PP für das *Terminieren der MAC Encryption*-Szenario

Sichtweise:	PP
Protokollszenario:	Terminieren der MAC Encryption
Spoofing:	—
Tampering:	• Ein Angreifer kann die übertragenen Daten manipulieren. [term-mac-encryption-manipulating-data]
Information Disclosure:	• Ein Angreifer kann unverschlüsselte Daten mitlesen. [term-mac-encryption-eavesdropping-data]
Denial of Service:	• Ein Angreifer stört die Kommunikation durch Jamming. [term-mac-encryption-jamming]

Tabelle 2.20: Threat Model aus Sicht des FP für das *Terminieren der MAC Encryption*-Szenario

Sichtweise:	FP
Protokollszenario:	Terminieren der MAC Encryption
Spoofing:	• Ein Angreifer kann eine `STOP.REQ`-Nachricht injizieren, sodass der MAC-Layer des FP aufhört, die Nachrichten zu verschlüsseln. [term-mac-encryption-injecting-stop-req]
Tampering:	• Ein Angreifer kann die übertragenen Daten manipulieren. [term-mac-encryption-manipulating-data] • Ein Angreifer kann ein A-Field einer PP-Nachricht so manipulieren, dass es die `STOP.REQ`-Nachricht beinhaltet. [term-mac-encryption-changing-to-stop-req]
Information Disclosure:	• Ein Angreifer kann unverschlüsselte Daten mitlesen. [term-mac-encryption-eavesdropping-data]
Denial of Service:	• Ein Angreifer stört die Kommunikation durch Jamming. [term-mac-encryption-jamming]

Schutzmechanismen

Gefahr: [term-mac-encryption-injecting-stop-req] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Die MAC-Encryption schützt grundsätzlich nicht ausreichend vor der Injektion oder Manipulation von Nachrichten, da die übertragenen Nachrichten nicht integritätsgesichert sind.

Da bei ULE zusätzlich zur MAC Encryption immer eine authentifizierte Verschlüsselung mit AES-CCM durchgeführt wird, sind auch nach einem von einem Angreifer ausgelösten Ende der MAC-Verschlüsselung keine Applikationsdaten lesbar.

Auswirkungen: Die MAC-Verschlüsselung wird beendet.

Gefahr: [term-mac-encryption-manipulating-data] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Die ULE Spezifikation sieht vor, dass alle Applikationsdaten zusätzlich auf dem DLC-Layer mithilfe von AES-CCM verschlüsselt werden. Es ist nicht vorgesehen, diese Verschlüsselung zu beenden. Ein Angreifer kann den von AES-CCM generierten Message Integrity Code nicht fälschen, ohne Kenntnis über den Authentication Key zu besitzen. Deshalb kann ein Angreifer auch nach Beendigung der MAC-Encryption keine Applikationsdaten manipulieren.

Gefahr: [term-mac-encryption-changing-to-stop-req] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Diese Gefahr ist sehr ähnlich zu der Gefahr [term-mac-encryption-injecting-stop-req].

Da bei ULE zusätzlich zur MAC Encryption immer eine authentifizierte Verschlüsselung mit AES-

CCM durchgeführt wird, sind auch nach einem von einem Angreiferausgelösten Ende der MAC-Verschlüsselung keine Applikationsdaten lesbar.

Auswirkungen: Die manipulierten Daten werden verworfen und vom Empfänger nicht bestätigt. Dies provoziert eine Retransmission der Daten.

Gefahr: [term-mac-encryption-eavesdropping-data]	verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Nachdem die Verschlüsselung beendet wird, werden vom MAC-Layer keine weiteren Daten verschlüsselt.

Die ULE Spezifikation sieht vor, dass alle Applikationsdaten zusätzlich auf dem DLC-Layer mithilfe von AES-CCM verschlüsselt werden. Es ist nicht vorgesehen, diese Verschlüsselung zu beenden. Deshalb können auch nach Beendigung der MAC-Encryption keine Applikationsdaten mitgelesen werden.

Gefahr: [term-mac-encryption-jamming]	Verfahren zur Schadensbegrenzung

Schutzmechanismus: Die MAC-Verschlüsselung wird nicht beendet. Dies ist mindestens für die initiierende Partei (den PP) zu erkennen, da die `STOP.CONF`-Nachricht nicht empfangen wird. Der PP kann die Prozedur erneut initiieren. Neben den Kontrollnachrichten der Terminierung der MAC-Encryption wird ggf. auch die Übertragung von Applikationsdaten gestört.

Auswirkungen: Der Empfang der Daten, die im B-Field übertragen werden sollten, wird vom Empfänger nicht bestätigt. Dies provoziert eine Retransmission der Daten.

2.7.9 Schlüsselwechsel der MAC Encryption

[EN 300 175-7, Abschnitt 6.4.6.5]

Der Schlüssel, der zur Verschlüsselung einer MAC Connection verwendet wird, kann ausgetauscht werden. Dies ist zum Beispiel beim Wechsel der *Early Encryption* zu einer Verschlüsselung mit einem *DCK* notwendig.

Zum Schlüsselwechsel wird zunächst die Verschlüsselung mit dem aktuellen Schlüssel beendet (siehe Abschnitt 2.7.8 „Terminieren der MAC Encryption“). Anstelle der Bestätigung der Terminierung der Verschlüsselung durch die Nachricht `STOP.GRANT` vom PP sendet dieser in diesem Szenario abweichend eine neue Aufforderung zur Verschlüsselung analog zum Szenario in Abschnitt 2.7.7 „Starten der MAC Encryption“.

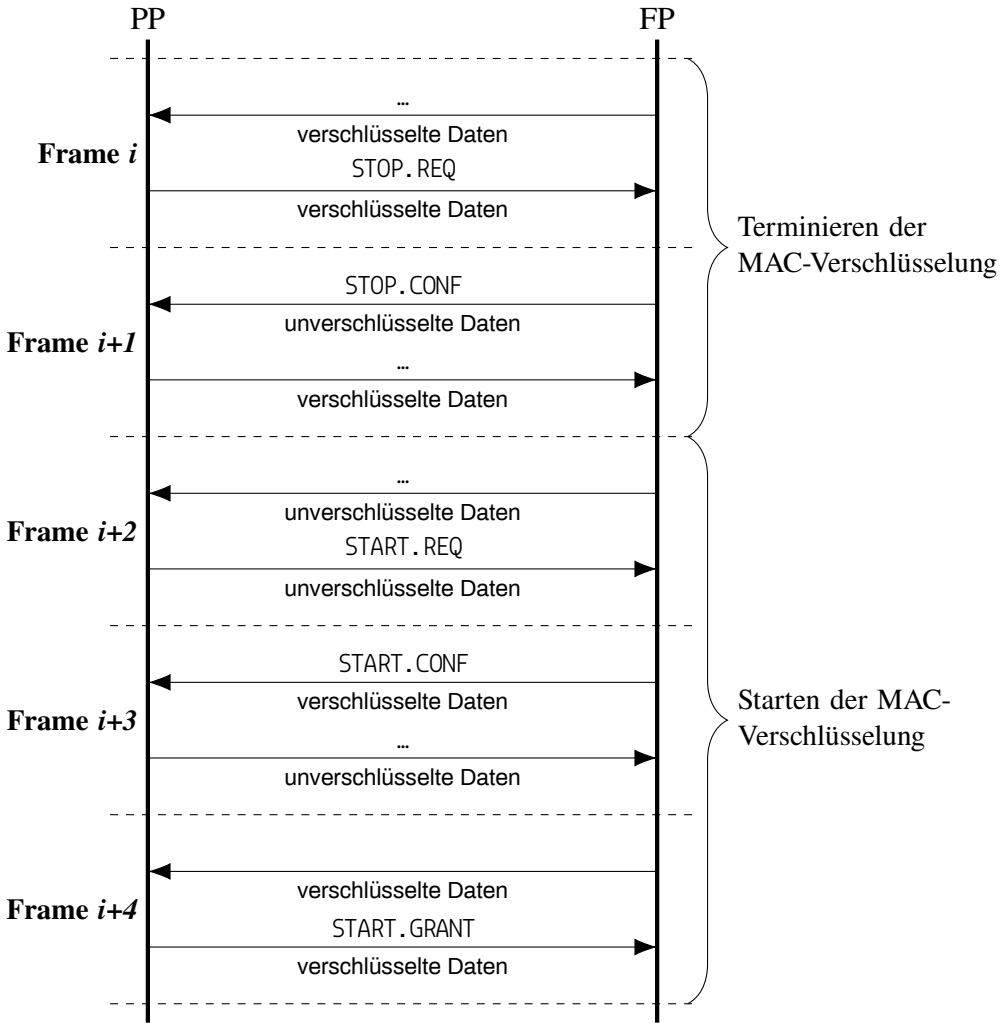


Abbildung 2.14: Idealer Nachrichtenfluss bei erfolgreichem Rekeying der MAC-Verschlüsselung

Threat Model

Tabelle 2.21: Threat Model aus Sicht des PP für das *Schlüsselwechsel der MAC Encryption*-Szenario

Sichtweise:	PP
Protokollszenario:	Schlüsselwechsel der MAC Encryption
Spoofing:	—
Tampering:	• Ein Angreifer manipuliert die übertragenen Daten. [rekeying-mac-encryption-manipulating-data]
Information Disclosure:	• Ein Angreifer liest unverschlüsselte Daten mit. [rekeying-mac-encryption-eavesdropping-data]
Denial of Service:	• Ein Angreifer sendet eine `START.REJECT`-Nachricht, um die Verschlüsselung zu unterbinden. [rekeying-mac-encryption-injecting-start-reject] • Ein Angreifer stört die Kommunikation durch Jamming. [rekeying-mac-encryption-jamming]

Tabelle 2.22: Threat Model aus Sicht des FP für das *Schlüsselwechsel der MAC Encryption*-Szenario

Sichtweise:	FP
Protokollszenario:	Schlüsselwechsel der MAC Encryption
Spoofing:	• Ein Angreifer kann eine `STOP.REQ`-Nachricht injizieren, sodass der MAC-Layer des FP aufhört, die Nachrichten zu verschlüsseln. [rekeying-mac-encryption-injecting-stop-req]
Tampering:	• Ein Angreifer manipuliert die übertragenen Daten. [rekeying-mac-encryption-manipulating-data]
Information Disclosure:	• Ein Angreifer liest unverschlüsselte Daten mit. [rekeying-mac-encryption-eavesdropping-data]
Denial of Service:	• Ein Angreifer stört die Kommunikation durch Jamming. [rekeying-mac-encryption-jamming]

Schutzmechanismen

Gefahr: [rekeying-mac-encryption-injecting-stop-req] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Nachdem der FP eine (injizierte) `STOP.REQ`-Nachricht empfängt, antwortet er mit einer `STOP.CONF`-Nachricht. Wenn der PP diese Nachricht enthält, kann dieser die MAC-Verschlüsselung erneut anfordern, indem er erneut eine `START.REQ`-Nachricht sendet.

In ULE Kommunikation wird die Kommunikation bereits auf dem DLC-Layer mit AES-CCM verschlüsselt.

Auswirkungen: Das Rekeying der MAC-Verbindung wird nicht vollständig durchgeführt und ggf. wird die MAC-Verschlüsselung beendet.

Gefahr: [rekeying-mac-encryption-manipulating-data] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Wenn ein Angreifer Daten manipuliert, kann dies nicht durch MAC-Verschlüsselung verhindert werden, da es sich dabei ausschließlich um ein „privacy feature“ handelt. Diese Gefahr ist identisch zu der Gefahr [start-mac-encryption-manipulate-data].

Dadurch, dass die Applikationsdaten bereits auf dem DLC-Layer mit einem Message Integrity Code ausgestattet werden, kann ein Angreifer die Daten nicht unbemerkt manipulieren.

Auswirkungen: Der Empfänger der Daten erkennt die Manipulation und verwirft die Daten, ohne deren Empfang zu bestätigen.

Gefahr: [rekeying-mac-encryption-eavesdropping-data] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Während des Schlüsselwechsels werden einige B-Fields nicht durch die MAC-Encryption verschlüsselt. Ein Angreifer kann den Klartext der Nutzdaten jedoch trotzdem nicht mitlesen, da dieser bei ULE zuvor mit AES-CCM verschlüsselt wird.

Bei DECT-Kommunikationen, in denen AES-CCM nicht eingesetzt wird, könnten die Geräte alternativ sicherstellen, keine unverschlüsselten Nutzdaten zu versenden – sondern mit dem Versand solange zu warten, bis die Verschlüsselung (wieder) aufgebaut ist.

Gefahr: [rekeying-mac-encryption-injecting-start-reject] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Das Rekeying scheitert, wenn ein Angreifer eine `START.REJECT`-Nachricht injiziert. Wenn der PP diese Nachricht empfängt, wird er keine `START.GRANT`-Nachricht versenden, sodass auch der FP das Scheitern bemerkt. In dem Status ist die MAC-Verschlüsselung jedoch bereits beendet. Der PP kann hier allerdings den Start der MAC-Verschlüsselung erneut initiieren (siehe Abschnitt 2.7.7 „Starten der MAC Encryption“).

Da die Nachrichten allerdings bereits auf dem DLC-Layer verschlüsselt und signiert werden, entsteht durch die fehlende MAC-Verschlüsselung keine Gefahr

Auswirkungen: Die MAC-Verschlüsselung wird nicht wieder aufgebaut.

Gefahr: [rekeying-mac-encryption-jamming] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Die MAC-Verschlüsselung wird ggf. nicht beendet oder nicht wieder gestartet, wenn die Kommunikation während des Rekeying gestört wird. Beide Fälle sind für die initiierende Partei (den PP) erkennbar, da entweder die `STOP.CONF`-Nachricht, oder die `START.CONF`-Nachricht nicht empfangen wird. In beiden Fällen kann der PP die Prozedur wieder initiieren. Im ersteren Fall kann der PP das gesamte Rekeying neu initiieren oder im zweiteren Fall die MAC-Verschlüsselung neu starten.

Wenn DECT Kommunikation nur auf MAC-Verschlüsselung setzt, kann die Datenübertragung so

lange pausiert werden, bis die Verschlüsselung wieder aktiv ist. Hierzu können die Applikationen beispielsweise die B-Fields der Nachrichten mit *Null-Messages* füllen.

Auswirkungen: Das Rekeying der MAC-Verbindung wird nicht vollständig durchgeführt und ggf. wird die MAC-Verschlüsselung beendet.

2.7.10 Permanent Virtual Circuit (PVC) Verwaltung

[TS 102 939-1, Abschnitt 12.1.3.1]

Wie in Abschnitt 2.4.5 „Network (NWK)“ beschrieben, ist für ULE ein vereinfachter NWK-Layer definiert. Indirekt wird direkt nach dem Pairing ein PVC aufgebaut. Dieser PVC ist entweder im Status *suspended* oder *resumed*, wobei er die Datenübertragung nur im Status *resumed* unterstützt. Beide Kommunikationspartner können den Status des PVC verändern, indem sie eine CC-SERVICE-CHANGE-Nachricht versenden. Innerhalb dieser Nachricht wird festgelegt, zu welchem Status gewechselt werden soll.

Resumption

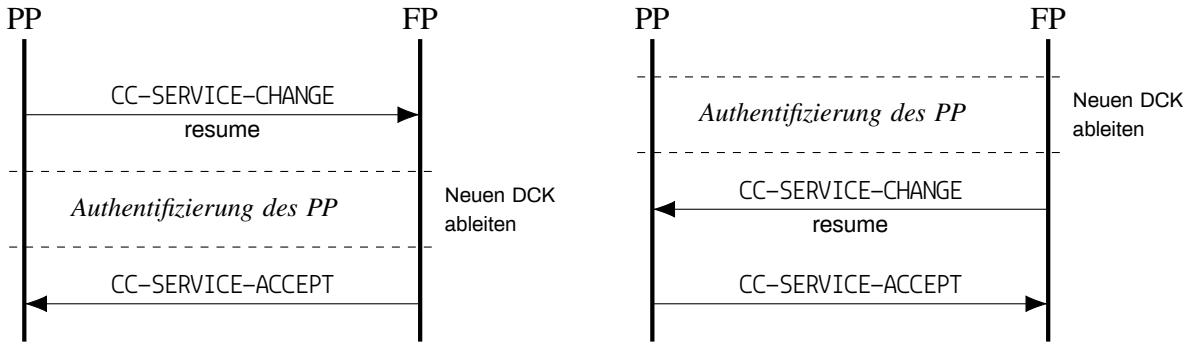
Bei der Resumption des PVC muss der FP in jedem Fall sicherstellen, dass ein bisher ungenutzter *DCK* zur Verfügung steht, bevor der Status von *suspended* zu *resumed* übergeht. Wenn der PP die CC-SERVICE-CHANGE-Nachricht sendet, um die Verbindung fortzusetzen (Statuswechsel zu *resumed*) muss der FP den PP zunächst erneut authentifizieren (siehe Abschnitt 2.7.3 „Authentifizierung eines PP“), um einen neuen *DCK* abzuleiten, bevor er diesen Statuswechsel akzeptiert. Falls bereits ein ungenutzter *DCK* zur Verfügung steht, kann auch dieser genutzt werden.

Erst wenn ein ungenutzter *DCK* zur Verfügung steht, antwortet der FP mit der CC-SERVICE-ACCEPT-Nachricht. Wenn kein *DCK* zur Verfügung steht und auch kein neuer generiert werden kann, bricht der FP den Verbindungsaufbau mit dem Versand der Nachricht CC-SERVICE-REJECT ab.

Nach dem Versand der CC-SERVICE-ACCEPT-Nachricht wird automatisch implizit der *LU14*-Link (siehe Abschnitt 2.4.4 „Data Link Control (DLC)“) aufgebaut. Alle Daten, die nach dieser Nachricht versendet werden, werden mit AES-128 im CCM Modus verschlüsselt und authentisiert.

Wenn der FP das Fortsetzen der Verbindung initiieren möchte, muss er, bevor er die CC-SERVICE-CHANGE-Nachricht versendet, sicherstellen, dass ein ungenutzter *DCK* zur Verfügung steht. Hierzu muss der FP in der Regel vorher das Szenario *Authentifizierung eines PP* durchführen.

In Abbildung 2.15 ist ein Beispiel für eine Kommunikation zur *Resumption* eines PVC zu finden.



(a) PP initiiert (b) FP initiiert

Abbildung 2.15: Nachrichtenfluss bei erfolgreicher PVC Resumption

Threat Model

Tabelle 2.23: Threat Model aus Sicht des PP für das *PVC Resumption*-Szenario

Sichtweise:	PP
Protokollszenario:	PVC Resumption
Spoofing:	• Ein Angreifer gibt sich als FP aus und initiiert eine Verbindung mit dem PP. [pvc-resumption-spoofing-fp] • Ein Angreifer gibt sich als FP aus, authentifiziert den PP und initiiert daraufhin eine Verbindung mit dem PP. [pvc-resumption-spoofing-fp-with-auth] • Ein Angreifer gibt sich als FP aus, während ein PP eine Verbindung initiiert. [pvc-resumption-pp-inited-spoofing-fp]
Tampering:	—
Information Disclosure:	—
Denial of Service:	• Ein Angreifer stört die Kommunikation durch Jamming. [pvc-resumption-jamming]

Tabelle 2.24: Threat Model aus Sicht des FP für das *PVC Resumption*-Szenario

Sichtweise:	FP
Protokollszenario:	PVC Resumption
Spoofing:	• Ein Angreifer spooft den PP, sodass der FP eine Verbindung mit dem Angreifer eingeht. [pvc-resumption-spoofing-pp]
Tampering:	—
Information Disclosure:	—
Denial of Service:	• Ein Angreifer stört die Kommunikation durch Jamming. [pvc-resumption-jamming]

Schutzmechanismen

Gefahr: [pvc-resumption-spoofing-fp] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Es ist nicht definiert, was passiert, wenn ein Angreifer den FP spooft und dabei keine Authentifizierung des PP durchführt.

Da allerdings in jedem Fall die nachfolgende Kommunikation verschlüsselt und signiert stattfindet, muss ein Angreifer hierzu Kenntnis über den verwendeten *DCK* haben. Es wird davon ausgegangen, dass ein Angreifer diesen geheimen Wert nicht kennt und somit auch eine etwaige Wiederverwendung eines alten *DCK* keine größeren Gefahren birgt, insbesondere wenn die Kommunikation frühzeitig beendet wird, da der Angreifer keine korrekten Nachrichten versenden kann.

Auswirkungen: Der PP geht eine Verbindung mit einem Angreifer ein, obwohl der Angreifer nicht in der Lage ist, dem PP valide signierte Kommandos zu schicken.

Gefahr: [pvc-resumption-spoofing-fp-with-auth] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Wenn ein Angreifer sich als FP ausgibt und die Authentifizierung des PP initiiert, ist dieser dabei nicht in der Lage, den korrekten *DCK* abzuleiten, der zur Verschlüsselung und Signierung der nachfolgenden Kommunikation genutzt werden muss. Um diesen *DCK* korrekt berechnen zu können, muss der FP den Authentication Key kennen.

Da ein Angreifer den Authentication Key nicht kennt, ist er nicht in der Lage, den Wert von *DCK* korrekt zu bestimmen. Er kann deshalb in der nachfolgenden Kommunikation weder Nachrichten senden noch lesen.

Auswirkungen: Der PP geht eine Verbindung mit einem Angreifer ein, obwohl der Angreifer nicht in der Lage ist, dem PP valide signierte Kommandos zu schicken.

Gefahr: [pvc-resumption-pp-inited-spoofing-fp] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Ein Angreifer kann sich während einer Verbindungsinitiierung eines PP nicht als FP ausgeben.

Ein Angreifer, der sich an die Spezifikation hält und die Authentifizierung des PP durchführt, um einen neuen *DCK* abzuleiten, ist nicht in der Lage den gleichen *DCK* wie der PP zu berechnen. Deshalb ist er danach nicht im Stande, die Nachrichten des PP zu entschlüsseln. Ferner kann er seine Nachrichten an den PP nicht authentisieren, sodass diese auf der Gegenseite verworfen werden.

Auch wenn sich der Angreifer nicht an die Spezifikation hält und keine Authentifizierung des PP durchführt, ist er nicht in der Lage, mit dem PP zu kommunizieren. Auch über bereits verwendete *DCK*s hat ein Angreifer keine Kenntnis. Aus diesem Grund kann der Angreifer auch in diesem Fall weder Nachrichten an den PP senden, noch Nachrichten von diesem entschlüsseln.

Auswirkungen: Der PP geht eine Verbindung mit einem Angreifer ein, obwohl der Angreifer nicht in der Lage ist, dem PP valide signierte Kommandos zu schicken.

Gefahr: [pvc-resumption-spoofing-pp]	verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Vor Beginn einer neuen Verbindung muss ein neuer *DCK* zur Verfügung stehen. Hierzu authentifiziert ein FP vor dem Versand der `CC-SERVICE-CHANGE`-Nachricht zunächst den PP. Alternativ – wenn bereits ein neuer *DCK* zur Verfügung steht – wird dieser für die nachfolgende Kommunikation genutzt.

Im ersten Fall, in dem der PP vor der `CC-SERVICE-CHANGE`-Nachricht vom FP authentifiziert wird, scheitert die Authentifizierung, weil der Angreifer nicht den korrekten Authentication Key kennt und der FP bricht den Verbindungsaufbau ab.

Im zweiten Fall, in dem bereits ein *DCK* zur Verfügung steht, kann ein Angreifer während der Verbindung weder Nachrichten senden, noch lesen. Er kann weder die Nachrichten entschlüsseln, noch Nachrichten verschlüsseln und mit einem korrekten Message Integrity Code ausstatten.

Gefahr: [pvc-resumption-jamming]	Verfahren zur Schadensbegrenzung

Schutzmechanismus: Wenn der Kommunikationskanal gestört ist, kann der PVC nicht fortgesetzt werden. Es kann keine MAC-Verbindung aufgebaut werden. In der Regel kann die Störung der Kommunikation allerdings früher erkannt werden.

a) Im Falle der FP-initiierten PVC-Resumption wird der FP bereits während der Authentifizierung des PP keine korrekte Antwort vom PP erhalten und so das Scheitern der PVC Resumption bemerken.

b) Im Falle der PP-initiierten PVC-Resumption wird der FP weder den PP authentifizieren noch die `CC-SERVICE-ACCEPT`-Nachricht versenden, um die Resumption zu bestätigen.

Auswirkungen: Der Empfänger der Daten erhält diese nicht und wird so auch deren Empfang nicht bestätigen. Durch ausbleibende Bestätigungen wird der Sender den Versand wiederholen.

Suspension

Wenn die Kommunikation beendet werden soll, wird der Status des PVC zu *suspended* geändert. Hierfür können beide Parteien die `CC-SERVICE-CHANGE`-Nachricht senden, die diesen Statuswechsel initiiert. Die Gegenseite quittiert dem Empfang dieser Nachricht mit einer `CC-SERVICE-ACCEPT`.

Nach erfolgreicher Suspension des PVC wird der zur Verbindung gehörige MAC-Bearer freigegeben.

Threat Model

Tabelle 2.25: Threat Model aus Sicht des PP für das *PVC Suspension*-Szenario

Sichtweise:	PP
Protokollszenario:	PVC Suspension
Spoofing:	• Ein Angreifer gibt sich als FP aus und beendet die Verbindung. [pvc-suspension-spoofing-fp]
Tampering:	—
Information Disclosure:	—
Denial of Service:	• Ein Angreifer stört die Übertragung der `CC-SERVICE-ACCEPT`-Nachricht. [pvc-suspension-jamming] • Ein Angreifer verändert die Nachricht `CC-SERVICE-ACCEPT` so, dass sie nach der Manipulation das Kommando `CC-SERVICE-REJECT` beinhaltet. [pvc-suspension-changing-service-accept]

Tabelle 2.26: Threat Model aus Sicht des FP für das *PVC Suspension*-Szenario

Sichtweise:	FP
Protokollszenario:	PVC Suspension
Spoofing:	• Ein Angreifer gibt vor, der PP zu sein und beendet die Verbindung. [pvc-suspension-spoofing-pp]
Tampering:	—
Information Disclosure:	—
Denial of Service:	—

Schutzmechanismen

Gefahr: [pvc-suspension-spoofing-fp] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Die Beendigung des PVC erfordert den Versand der `CC-SERVICE-CHANGE`-Nachricht. Da diese Nachricht vom NWK-Layer an den DLC-Layer übergeben wird, verschlüsselt und authentisiert dieser die Nachricht mithilfe von AES-CCM.

Ein Angreifer kann diese Nachricht nicht im Namen des FP signieren und verschlüsseln.

Gefahr: [pvc-suspension-spoofing-pp] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Diese Gefahr ist ähnlich zu [pvc-suspension-spoofing-fp]. Auch aus Sicht des FP müssen alle Nachrichten, die innerhalb eines PVC versendet werden, verschlüsselt und authentisiert sein. Falsch authentisierte Nachrichten (auch `CC-SERVICE-CHANGE`-Nachrichten) werden nicht verarbeitet.

Gefahr: [pvc-suspension-jamming]	verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Alle Nachrichten, die während des PVC-resumed Status versandt werden, werden verschlüsselt und authentisiert. Ein Angreifer ist nicht in der Lage herauszufinden, welche Nachricht die CC-SERVICE-ACCEPT-Nachricht ist.

Gefahr: [pvc-suspension-changing-service-accept]	verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Alle Nachrichten, die während des PVC-resumed Status versandt werden, werden verschlüsselt und authentisiert. Ein Angreifer ist nicht in der Lage, eine Nachricht unbemerkt zu verändern, ohne den Wert des Authentication Key zu kennen.

2.7.11 Connection Handover

[EN 300 444, Abschnitt 9.7]

Wenn z. B. die Qualität einer Verbindung schlechter wird, kann der DLC-Layer entscheiden, eine neue MAC-Verbindung auf einem anderen Funkkanal aufzubauen. Auf diese neue Verbindung wird die bestehende mithilfe des *Connection Handover*s übergeleitet. Dieses Verfahren ist für ULE Geräte optional.

Um einen Connection Handover durchzuführen, wird zunächst parallel zur bestehenden MAC-Verbindung eine neue Verbindung aufgebaut. Hierbei wird die neue MAC-Verbindung – wie jede andere MAC-Verbindung – entweder mit MAC *Early Encryption* oder zunächst ohne MAC-Verschlüsselung aufgebaut. Während beide Verbindungen gleichzeitig aufgebaut sind, werden alle Daten vom DLC-Layer über beide Verbindungen versandt. Um die MAC-Verschlüsselung auf der neuen Verbindung zu aktivieren, muss abermals das *Authentifizierung eines PP*-Szenario durchgeführt werden, um einen neuen Schlüssel für die neue Verbindung abzuleiten. Mit diesem neuen – vom alten Schlüssel unabhängigen – Schlüssel kann dann die neue Verbindung durch den MAC-Layer verschlüsselt werden. Sobald beide Verbindungen verschlüsselt sind, wird die alte Verbindung abgebaut. Wenn die alte Verbindung abgebaut ist, fließen alle Daten ausschließlich über die neue Verbindung und der Handover wurde erfolgreich durchgeführt.

Threat Model

Tabelle 2.27: Threat Model aus Sicht des PP für das *Connection Handover*-Szenario

Sichtweise:	PP
Protokollszenario:	Connection Handover
Spoofing:	• Ein Angreifer gibt sich bei der zweiten Verbindung zum PP als FP aus, sodass der PP nach dem Handover Nachrichten vom Angreifer empfängt. [connection-handover-spoofing-fp]
Tampering:	• Ein Angreifer manipuliert die übertragenen Daten. [connection-handover-manipulating-data]
Information Disclosure:	• Ein Angreifer liest unverschlüsselte Daten während des Handovers mit. [connection-handover-eavesdropping-data]
Denial of Service:	• Ein Angreifer verändert die auf der neuen Verbindung übertragenen Daten, um den Handover scheitern zu lassen. [connection-handover-changing-data] • Ein Angreifer stört den Handover durch Jamming. [connection-handover-jamming]

Tabelle 2.28: Threat Model aus Sicht des FP für das *Connection Handover*-Szenario

Sichtweise:	FP
Protokollszenario:	Connection Handover
Spoofing:	• Ein Angreifer gibt sich bei der zweiten Verbindung zum FP als PP aus, sodass der FP nach dem Handover Nachrichten vom Angreifer empfängt. [connection-handover-spoofing-pp]
Tampering:	• Ein Angreifer manipuliert die übertragenen Daten. [connection-handover-manipulating-data]
Information Disclosure:	• Ein Angreifer liest unverschlüsselte Daten während des Handovers mit. [connection-handover-eavesdropping-data]
Denial of Service:	• Ein Angreifer verändert die auf der neuen Verbindung übertragenen Daten, um den Handover scheitern zu lassen. [connection-handover-changing-data] • Ein Angreifer stört den Handover durch Jamming. [connection-handover-jamming]

Schutzmechanismen

Gefahr: [connection-handover-spoofing-fp] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Die Applikationsdaten, die über die Verbindung versendet werden, sind bei ULE in jedem Fall verschlüsselt und authentisiert. Hierzu wird der Algorithmus AES-CCM verwendet. Damit ein Angreifer die neue Verbindung unbemerkt übernehmen kann, müsste er in der Lage sein, im Namen des FP die Daten zu signieren. Dies ist nicht der Fall, da ein Angreifer den Wert von *DCK* nicht kennt.

Auswirkungen: Das Connection Handover scheitert und die bestehende Verbindung wird weiter verwendet.

Gefahr: [connection-handover-spoofing-pp] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Ein FP kann erkennen, wenn Nachrichten nicht korrekt vom legitimen PP empfangen werden. Diese Gefahr ist sehr ähnlich zu [handover-spoofing-fp]. Um den PP erfolgreich fälschen zu können, muss der Angreifer Nachrichten an den FP mit dem ihm unbekannten *DCK* verschlüsseln und signieren.

Auswirkungen: Das Connection Handover scheitert und die bestehende Verbindung wird weiter verwendet.

Gefahr: [connection-handover-manipulating-data] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Die übertragenen Applikationsdaten werden vom DLC-Layer authentifiziert und verschlüsselt. Der Empfänger überprüft bei empfangenen Daten zunächst, ob die Signatur korrekt ist.

Da ein Angreifer manipulierte Daten nicht korrekt signieren kann, wird diese Überprüfung beim Empfänger fehlschlagen und die Daten werden nicht verarbeitet.

Auswirkungen: Das Connection Handover scheitert und die bestehende Verbindung wird weiter verwendet.

Gefahr: [connection-handover-eavesdropping-data] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Obwohl beim Handover die Daten zeitweise nicht durch den MAC-Layer verschlüsselt werden, ist ein Angreifer bei einer ULE-Verbindung nicht in der Lage, den Inhalt der Nachrichten mitzulesen. Die Applikationsdaten werden auf dem DLC-Layer bereits mit AES-CCM verschlüsselt. Diese Verschlüsselung wird beim Handover nicht unterbrochen.

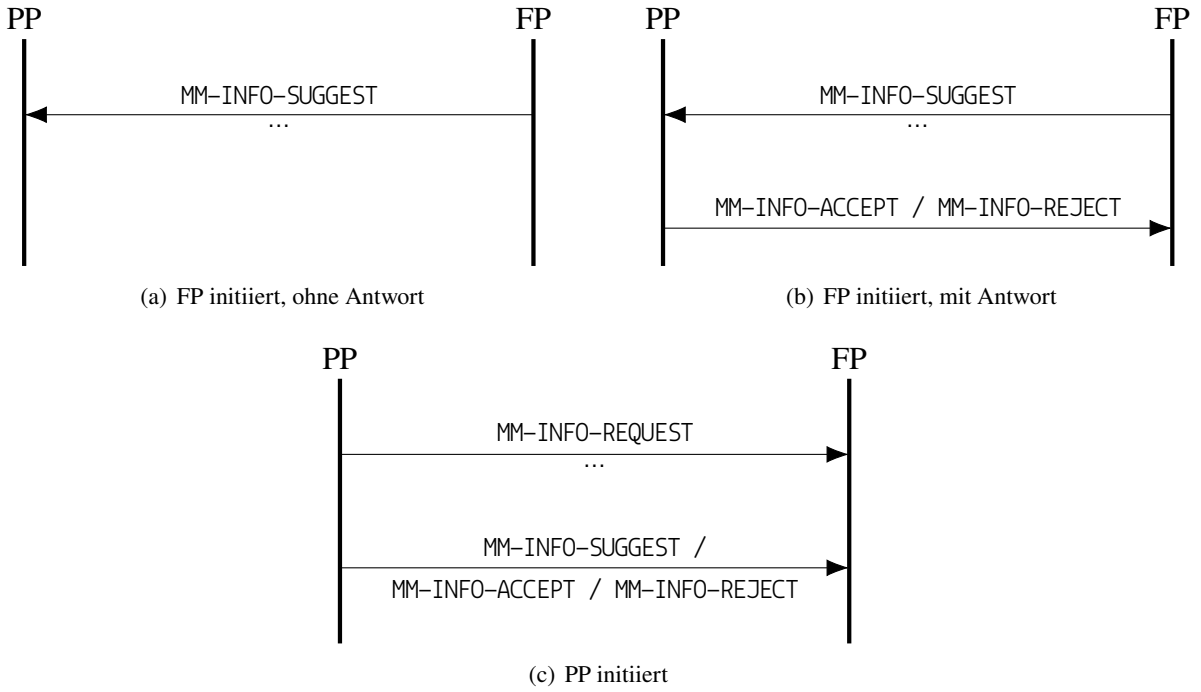
Gefahr: [connection-handover-changing-data] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Wenn die Daten auf der neuen Verbindung manipuliert werden, wird der Handover scheitern, da ein Partner bemerkt, dass sich die ausgetauschten Daten auf beiden Verbindungen unterscheiden.

Auswirkungen: Das Connection Handover scheitert und die bestehende Verbindung wird weiter verwendet.

Gefahr: [connection-handover-jamming] | Verfahren zur Schadensbegrenzung

Schutzmechanismus: Wenn die Kommunikation gestört wird, sind die Daten, die auf beiden Verbindungen empfangen werden, nicht identisch. Wenn dies erkannt wird, scheitert der Handover. In diesem Fall wird die bestehende Verbindung weiterhin verwendet.



(a) FP initiiert, ohne Antwort

(b) FP initiiert, mit Antwort

(c) PP initiiert

Abbildung 2.16: Nachrichtenfluss des *Parameter Retrieval*

Auswirkungen: Das Connection Handover scheitert und die bestehende Verbindung wird weiter verwendet.

2.7.12 Parameter Retrieval

[EN 300 175-5, Abschnitt 13.7]

DECT bietet mit dem *Parameter Retrieval* ein Protokollszenario an, bei dem bestimmte Parameter (wie z. B. Schlüsselmaterial) an andere Geräte weitergegeben werden können. Dieses Szenario ist in drei Variationen definiert:

(a) FP initiiert, ohne Antwort

(b) FP initiiert, mit Antwort

(c) PP initiiert

Am Beispiel von zwei Verwendungen wird dieses Szenario beschrieben:

Schlüsselaustausch für Multicastgruppen

[EN 300 175-7, Abschnitt 6.3.8]

Auch Multicast-Nachrichten können verschlüsselt werden. Hierzu generiert der FP einen zufälligen Schlüssel. Mit diesem Schlüssel verschlüsselt der FP alle an die Multicastgruppe gerichteten Nachrichten. Um es den Multicastgruppen-Teilnehmern zu ermöglichen, die Nachrichten zu entschlüsseln, muss der FP den Schlüssel an jeden Multicastgruppen-Teilnehmer senden.

Für dieses Szenario wird entweder Variante (b) oder (c) des *Parameter Retrieval*-Szenarios verwendet.

Im Falle der Variante (b) sendet der FP in der `MM-INFO-SUGGEST`-Nachricht eine Identifikation, um welche Informationen es sich handelt (*Cipher Key for CCM encryption of multicast channels* und *CCM Sequence number for CCM encryption of multicast channels*), die Multicastgruppen-Nummer, den CCM-Schlüssel und die CCM-Sequenznummer an den PP. Bei Empfang dieser Nachricht quittiert der PP dies mit der `MM-INFO-ACCEPT`-Nachricht.

Bei der PP-initiierten Variante dieses Szenarios (Variante (c)), fragt der PP mit der `MM-INFO-REQUEST`-Nachricht nach dem *Cipher Key for CCM encryption of multicast channels* und nach der *CCM Sequence number for CCM encryption of multicast channels* und gibt dazu die Multicastgruppen-Nummer an. Daraufhin antwortet der FP mit der `MM-INFO-SUGGEST`-Nachricht mit denselben Inhalten wie in Variante (b).

Um die Sicherheit der Daten zu gewährleisten, wird diese Prozedur immer über eine verschlüsselte Verbindung durchgeführt.

Übertragung eines *DCK* an eine WRS

[EN 300 175-7, Abschnitt 6.3.9]

Eine WRS (Wireless Relay Station) entschlüsselt die Daten eines PP auf MAC-Layer Ebene, um sie zum Weiterversand an den FP erneut verschlüsseln zu können. Hierzu muss sie den *DCK* der betreffenden Verbindung zwischen dem PP und dem FP erhalten. Auch dies wird mithilfe des *Parameter Retrieval*-Szenarios durchgeführt.

Diese Prozedur wird in der Regel vom FP initiiert (Variante (a)) – falls der PP jedoch Probleme beim Empfang hatte, kann er den Versand der Informationen auch noch einmal anfordern (Variante (c)).

Im Normalfall sendet der FP die Nachricht `MM-INFO-SUGGEST` an die WRS und definiert darin die Art des übertragenen Schlüssels und das eigentliche Schlüsselmaterial.

Wenn die WRS den erneuten Versand der Daten anfordern möchte, weil der vorherige Versand beispielsweise fehlgeschlagen ist, sendet sie eine `MM-INFO-REQUEST`-Nachricht an den FP, in der die WRS definiert, welchen Key (Sequenznummer oder DCK) sie erhalten möchte. Der FP antwortet dabei mit derselben Nachricht wie bei der FP-initiierten Variante dieses Szenarios.

Threat Model

Tabelle 2.29: Threat Model aus Sicht des PP für das *Parameter Retrieval*-Szenario

Sichtweise:	PP
Protokollszenario:	Parameter Retrieval
Spoofing:	• Ein Angreifer gibt sich als der Kommunikationspartner aus und sendet dem PP Daten. [parameter-retrieval-spoofing]
Tampering:	• Ein Angreifer manipuliert die übertragenen Werte. [parameter-retrieval-manipulating-values]
Information Disclosure:	• Ein Angreifer liest die übertragenen Daten mit. [parameter-retrieval-eavesdropping]
Denial of Service:	• Ein Angreifer stört die Kommunikation durch Jamming. [parameter-retrieval-jamming]

Tabelle 2.30: Threat Model aus Sicht des FP für das *Parameter Retrieval*-Szenario

Sichtweise:	FP
Protokollszenario:	Parameter Retrieval
Spoofing:	• Ein Angreifer gibt sich als der Kommunikationspartner aus und sendet dem PP Daten. [parameter-retrieval-spoofing]
Tampering:	• Ein Angreifer manipuliert die übertragenen Werte. [parameter-retrieval-manipulating-values]
Information Disclosure:	• Ein Angreifer liest die übertragenen Daten mit. [parameter-retrieval-eavesdropping]
Denial of Service:	• Ein Angreifer stört die Kommunikation durch Jamming. [parameter-retrieval-jamming]

Schutzmechanismen

Gefahr: [parameter-retrieval-spoofing] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Bei ULE werden alle Daten mittels AES-CCM verschlüsselt und signiert. Ein Angreifer ist demnach nicht in der Lage, Nachrichten im Namen eines legitimen Kommunikationsteilnehmers zu versenden. Er kann die Daten nicht korrekt signieren.

Auswirkungen: Der PP verwirft die Daten.

Gefahr: [parameter-retrieval-manipulating-values] | verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Bei ULE werden alle Daten mittels AES-CCM verschlüsselt und signiert. Ein Angreifer ist demnach nicht in der Lage, manipulierte Nachrichten korrekt zu signieren. Ein Empfänger wird fehlerhaft signierte Nachrichten nicht verarbeiten.

Gefahr: [parameter-retrieval-eavesdropping]	verpflichtendes Verfahren zur Abwehr

Schutzmechanismus: Bei ULE werden alle Daten mittels AES-CCM verschlüsselt und signiert. Ein Angreifer ist demnach nicht in der Lage, die übertragenen Daten mitzulesen.

Auch bei anderen DECT Kommunikationen wird beim *Parameter Retrieval* von einigen Daten verlangt, dass die Verbindung, über die das Szenario ausgeführt wird, verschlüsselt ist. Ein Beispiel hierfür ist die Übertragung des *DCK* an WRS.

Gefahr: [parameter-retrieval-jamming]	Verfahren zur Schadensbegrenzung

Schutzmechanismus: Wenn der Kommunikationskanal gestört ist, werden die Daten nicht korrekt beim PP ankommen. Weder CRC-Summen, die der MAC-Layer zu den Nachrichten hinzufügt, noch die CCM Signatur des DLC Layers sind in diesem Fall korrekt. Wenn die Daten nicht korrekt übertragen werden, kann der PP sie mit der `MM-INFO-REQUEST`-Nachricht erneut anfragen.

3 Machbarkeitsstudie

In der theoretischen Analyse des ULE Protokolls wurden einige Gefahren erkannt zu denen kein ausreichender Schutz definiert ist. In diesem Teil der Arbeit wird nun überprüft, welchen Aufwand eine Durchführung von Angriffen gegen ULE darstellt. Insbesondere wird hierzu keine spezielle DECT Hardware verwendet, um eine Aussage darüber treffen zu können, ob diese Angriffe auch ohne professionelle Vorbereitung von Laien durchgeführt werden können. Ferner werden nur quelloffene Programme verwendet, um die Kosten für einen Angriff gegen das Protokoll gering zu halten.

Ziel der Machbarkeitsstudie ist demnach eine Überprüfung, inwiefern ULE – insbesondere aufbauend auf einer TDMA Struktur – mit universeller Funk-Hardware, sogenannten Software Defined Radios, passiv angegriffen werden kann. Zudem wird evaluiert, was darüber hinaus notwendig ist, um auch aktive Angriffe gegen ULE durchführen zu können.

3.1 Testumgebung

Die Testumgebung für die Machbarkeitsstudie besteht aus folgenden Geräten:

1. Panasonic KX-HN6011GWA-K

   Bei der Panasonic KX-HN6011GWA-K handelt es sich um eine Alarmanlage. Von Alarmanlagen erwartet ein Benutzer ein größtmögliches Maß an Sicherheit, weshalb Schwachstellen an diesem Gerät besonders sicherheitskritisch sind. Die Alarmanlage besteht aus vier verschiedenen Komponenten:

   - Basisstation
   - Fenstersensor
   - Bewegungsmelder
   - Sirene

   Der Hersteller gibt auf seiner Webseite an, dass die Geräte „DECT ULE kompatibel“ (siehe [PanasonicKX-HN6011]) sind.

2. AVM FRITZ!Box 7430

   Neben einem sicherheitsrelevanten System wie einer Alarmanlage wurde ein Gerät mit hoher Verbreitung gesucht, das ULE-fähig ist. Hier fiel die Wahl auf ein FRITZ!Box-Modell. Bereits 2010 lag der Marktanteil der Router von AVM in Deutschland bei 68 % (vgl. [Lücke2010]). Es ist davon

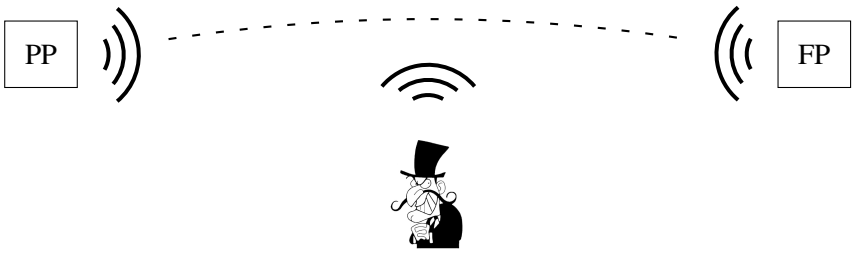


Abbildung 3.1: Testumgebung

auszugehen, dass durch die Entscheidung „zur Auswahl und zum Anschluss von Telekommunikationsendgeräten“ (siehe [Telkogeräte2016])[1] im Jahr 2016 der Marktanteil von AVM Routern in Privathaushalten noch weiter zugenommen hat.

Die meisten aktuellen FRITZ!Box-Geräte unterstützen DECT ULE.

3. AVM FRITZ!DECT 301 Heizkörperregler

   Als Gegenstück zur ULE Kommunikation der FRITZ!Box wurde eine zweite Hardware von AVM ausgewählt. Hier fiel die Wahl auf den Heizkörperregler AVM DECT 301.

In dieser Arbeit wird die Kommunikation zwischen einem PP und einem FP durchgeführt und abgehört. Der Fokus liegt auf einem passiven Angriff, sodass die direkte Kommunikation zwischen beiden Partnern nicht unterbrochen wird. Der Testaufbau ist in Abbildung 3.1 skizziert. Im Fazit wird evaluiert, was zusätzlich zu den hier durchgeführten Angriffen notwendig wäre, um die Verbindung zwischen beiden Teilnehmern gezielt zu stören.

Um die Funkverbindung abzuhören wird mit dem HackRF One von Great Scott Gadgets ein Software Defined Radio (SDR) eingesetzt. Ein Software Defined Radio lässt sich als eine Antenne beschreiben, die vollständig durch Software zu parametrisieren ist und bei der das empfangene Signal als Zahlenreihe von einem Hostcomputer ausgelesen werden kann. Mit einem SDR lässt sich ein Empfänger und Sender für beliebige Funkprotokolle entwickeln, da die SDR-Hardware keine protokollspezifischen Anpassungen enthält. Geräte, die Funk einsetzen, implementieren normalerweise einige Funktionen (z. B. Filter, Modulation des Signals, Synchronisierung, etc.) in protokollspezifischer Hardware und können so nur für ein bestimmtes Protokoll eingesetzt werden. Bei SDRs werden diese Funktionen vollständig in Software implementiert – daher auch der Name *Software Defined Radio*.

Als Hilfsmittel zur Auswertung des über das SDR empfangene Signal hat sich die Plattform *GNU Radio* etabliert. GNU Radio bietet eine Vielzahl an Bibliotheksfunktionen zur Signalverarbeitung an, mit denen Funksignale dekodiert werden können. Um diese Bibliotheksfunktionen einfach verschalten und weite Teile der Implementation vieler Protokolle sogar grafisch durchführen zu können, ist ein grafisches Werkzeug mit dem Namen *GNURadioCompanion* für GNU Radio entwickelt worden. Anhand von Datenflussgraphen können Signale von verschiedenen verknüpften *Blöcken* (Signalverarbeitungsfunktionen) verarbeitet werden. Es lassen sich verschiedenste SDRs als Signalquelle hinterlegen und Funktionen wie Tiefpassfilter oder Resampler können als *Block* in einem GNURadioCompanion Flowgraph eingefügt werden. Ein Beispiel für einen GNURadioCompanion Flowgraph findet sich in Abbildung 3.3 „Flowgraph der ReDECTed Implementation von DECT in GNURadioCompanion“.

[1] Diese Entscheidung wird häufig als Abschaffung des sogenannten *Routerzwangs* bezeichnet

3.2 Implementation

Um ULE Funkkommunikation mit einem SDR mitlesen und analysieren zu können, müssen alle in Abschnitt 2.4 beschriebenen Layer von ULE implementiert werden. Da die unteren Protokollschichten (PHL und MAC) von ULE identisch mit den Schichten von DECT sind, bietet es sich an, bestehende SDR-Programme für DECT zu nutzen und diese für die Verarbeitung von ULE anzupassen. Im Rahmen dieser Arbeit wurden insbesondere die beiden Projekte *gr-dect2* (siehe [gr-dect2]) und *ReDECTed* (siehe [ReDECTed]) evaluiert und angepasst.

3.2.1 ReDECTed

ReDECTed ist eine Implementation eines DECT-Sniffers für SDRs auf Basis von GNU Radio. Ausschließlich mit Standardfunktionen von GNU Radio demoduliert die Software das Funksignal und sendet dies mithilfe des *UDP Sink*-Blocks an einen lokalen UDP Socket. An diesem UDP Socket liest ein Programm die demodulierten Rohdaten und führt die Synchronisierung anhand der Synchronisationswörter durch (siehe Abschnitt 2.4.2 „Physical Layer (PHL)“). Wenn ein Zeitschlitz belegt ist – also ein Synchronisationswort erkannt wurde – erzeugt dieses Programm ein Ethernet-Paket und sendet darin die empfangenen Daten zur weiteren Verarbeitung an ein lokales Dummy-Interface.

Im deDECTed Projekt wurde bereits ein einfacher DECT-Dissector für Wireshark implementiert, der Pakete mit dem Ethertype `0x2323` als DECT markiert und grundlegend parst. Hierzu wurde ein separater Header für DECT Pakete oberhalb von Ethernet definiert, der Metadaten von mitgelesenem DECT Funkverkehr beinhaltet. Dieselbe Struktur verwendet auch ReDECTed, um die Nutzung desvorhandenen Dissectors von Wireshark zu unterstützen. Der Header hat das folgende Format:

`trxmode` Ein Flag, das angibt, ob das Paket gesendet oder empfangen wurde

`channel` Der DECT-Channel, auf dem das Paket gesendet oder empfangen wurde

`slot` Die Slotnummer des Pakets

`frameno` Die Nummer des Frames, in dem das Paket gesendet oder empfangen wurde

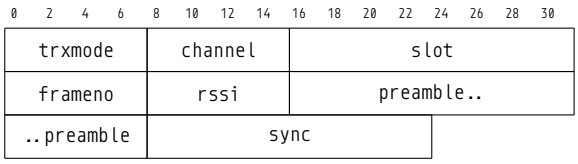


Abbildung 3.2: DECT-Header für Wireshark

`rssi` Der *Radio Signal Strength Indicator*, ein Wert, der die Qualität der Verbindung angibt

`preamble` Drei Byte Daten, die vor dem Synchronisationswort versendet werden (werden bei ULE nicht verwendet)

`sync` Die letzten zwei Bytes des Synchronisationsworts, ein Wert, aus dem geschlossen werden kann, ob das Paket von einem PP oder FP versandt wurde

Leider befüllt die ReDECTed Implementation ausschließlich die Werte `channel`, `preamble` und `sync` und verwaltet beispielsweise die Framenummer nicht. Durch die fehlende Framenummer ist eine tiefergehende Analyse der Daten sehr komplex oder gar unmöglich, da das B-Field nicht korrekt descrambled werden kann (siehe Abschnitt 2.4.3 „Media Access Control (MAC)“).

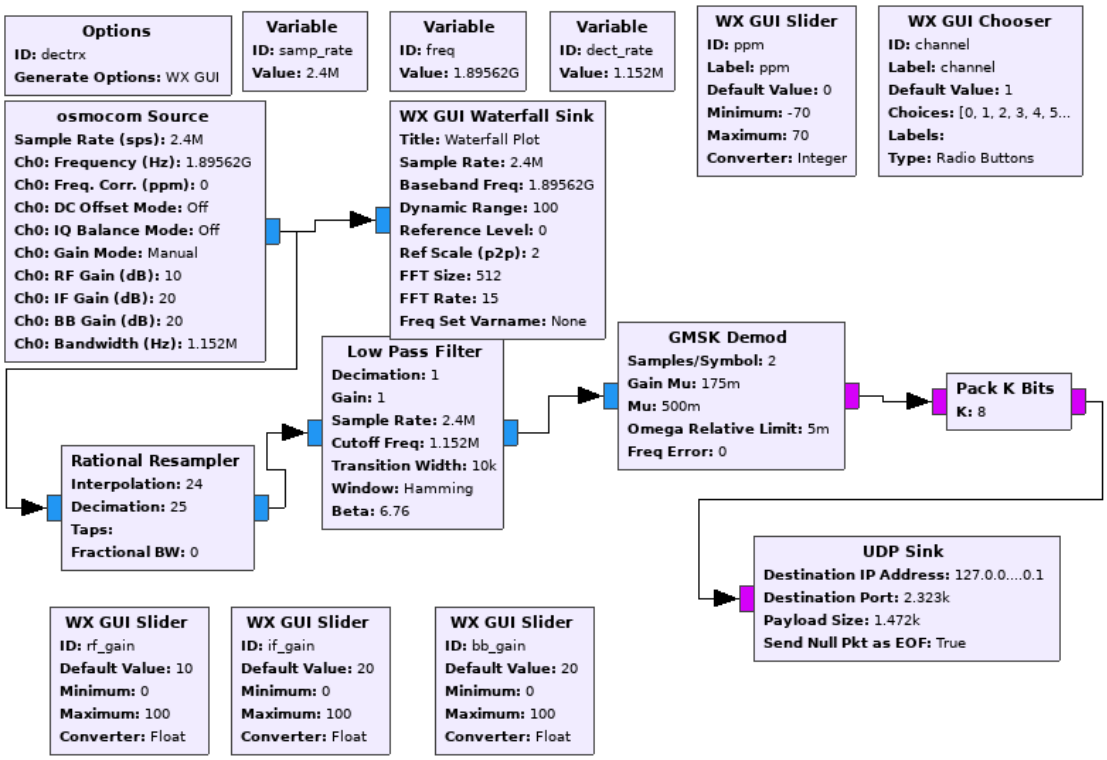


Abbildung 3.3: Flowgraph der ReDECTed Implementation von DECT in GNURadioCompanion

3.2.2 gr-dect2

Das Projekt *gr-dect2* wurde entwickelt, um den Voice Channel von DECT Telefonaten auf Basis von SDR-Daten abzuhören. Auch dieses Projekt nutzt zur Umsetzung GNU Radio. Anders als ReDECTed implementiert gr-dect2 eigene Blöcke für GNURadioCompanion und implementiert so eine Lösung ohne separate Anwendung, die den Datenstrom über einen UDP Socket empfängt und analysiert. Hierzu werden drei *Custom Blocks* definiert:

a) Der Block *Phase Diff* demoduliert das Signal.

b) Der Block *Packet receiver* sucht im Datenstrom nach den Synchronisationswörtern, verwaltet die Framenummern und implementiert die Unterscheidung verschiedener Bearer.

c) Der Block *Packet decoder* dekodiert die Pakete. Hier dekodiert er das A-Field um zu überprüfen, ob es sich um Voice-Daten handelt, descrambled das B-Field und bereitet die Daten so auf, dass sie im Nachgang an einen Audio-Decoder übergeben werden können, der die Audiodaten dann an die Soundkarte übergibt.

gr-dect2 ist – wie der *Packet decoder*-Block zeigt – speziell auf das Abhören von Telefonaten optimiert. Die Software bietet demnach auch keine Möglichkeit der Speicherung der Rohdaten des DECT-Verkehrs an, was eine nachträgliche Analyse von ULE Traffic unmöglich macht.

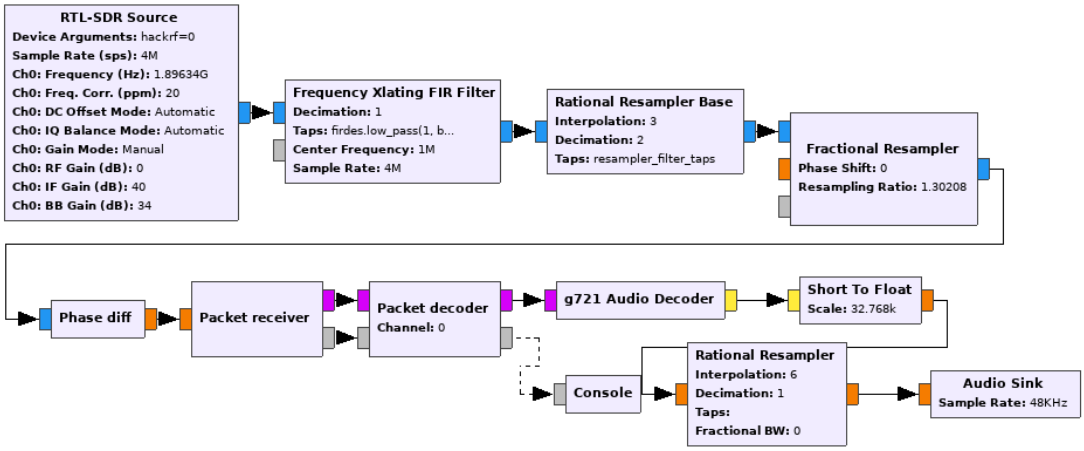


Abbildung 3.4: Ausschnitt des Flowgraphs von gr-dect2 in GNURadioCompanion

3.2.3 Anpassungen

Durch eine Kombination der beiden bestehenden Lösungen – ReDECTed und gr-dect2 –, lässt sich ein Framework entwickeln, mit dem ULE Verkehr vollumfänglich offline analysiert werden kann. Bei gr-dect2 fehlt dabei zunächst die Möglichkeit die dekodierten Daten zu speichern, um sie offline analysieren zu können. Hierzu empfiehlt sich der Ansatz, der durch das ReDECTed Projekt implementiert ist: den Versand der ULE-Daten in Ethernet-Frames an ein Dummy-Netzwerkinterface, um sie mithilfe von Wireshark aufnehmen und abspeichern zu können.

Gleichzeitig bietet es sich an, alle DECT-Channels gleichzeitig aufzuzeichnen, um auch in einer Offline-Datenanalyse auf Kanalwechsel reagieren zu können, ohne das SDR neu parametrisieren zu müssen. Während der Entwicklung ist aufgefallen, dass die verwendete Hardware nicht potent genug ist, um die Signalverarbeitung inklusive der DECT-Synchronisierung vollständig in Realzeit in Software durchzuführen. Deshalb werden zunächst die Funk-Rohdaten gespeichert. Auf diesen Daten lässt sich die Signalverarbeitung nachträglich durchführen, ohne dass die dazu verwendeten Programme Realzeitanforderungen erfüllen müssen.

Daraus lassen sich grundlegend vier Anpassungen für die gr-dect2 Software ableiten, die im Rahmen dieser Arbeit umgesetzt wurden:

i) Veränderte Parametrisierung des SDR, sodass alle DECT-Channels in der Bandbreite liegen und ausgewertet werden können

ii) Speicherung der Funk-Rohdaten und Änderung der Signalverarbeitung dahingehend, dass sie vormals gespeicherte Rohdaten verarbeiten kann

iii) Aufteilen des Signals in einzelne DECT Channels

iv) Versand des DECT-Payloads an ein Dummy-Interface

Parametrisierung des Software Defined Radio

Der HackRF One kann ein Signal mit einer maximalen Bandbreite von 20 MHz um eine eingestellte Basisfrequenz empfangen. Die maximale Samplerate beträgt 20 000 000 Samples/s, wobei jedes Sample mit einer Genauigkeit von 8 bit aufgelöst ist. Da DECT Channels jeweils Breite von 1,728 MHz besitzen, können alle zehn Channels im Frequenzbereich von 1880-1900 MHz gleichzeitig aufgenommen werden.

Abbildung 3.5: Der HackRF One

Zunächst muss die Basisfrequenz des HackRF festgelegt werden. Hierzu sollte eine Frequenz in der Mitte des verwendeten DECT-Frequenzbandes gewählt werden, um sicherzustellen, dass alle Channels in der Bandbreite des HackRF liegen. Bei dem Channel 0 handelt es sich um den Channel mit der höchsten Basisfrequenz (1897,344 MHz) und bei dem Channel 9 (1881,792 MHz) um den Channel mit der niedrigsten zu empfangenen Basisfrequenz. Aus diesem Grund empfiehlt es sich, den Mittelwert 1889,568 MHz beider Frequenzen als Basisfrequenz des HackRF zu nutzen.

Das Spektrum des Signals, das der HackRF One liefert, wird durch das Einstellen der Basisfrequenz automatisch um eben diese Frequenz verschoben, und als komplexwertiges IQ-Signal ausgegeben. Somit wird das empfangene Signal in ein sogenanntes *Basisband* mit der Basisfrequenz 0 Hz verschoben. Folglich haben alle realen Frequenzen der DECT-Channels eine Entsprechung im *heruntergemischten* HackRF One-Signal: die Frequenz der Differenz der realen Channel-Frequenz zur in der Hardware eingestellten Basisfrequenz 1889,568 MHz.

Um nun die im Basisband befindlichen Signale digital verarbeiten zu können, müssen sie vom HackRF One abgetastet und als digitale Samples an den Computer weitergegeben werden. Nach dem Nyquist-Shannon-Abtasttheorem muss die Samplerate eines abgetasteten Signals mindestens doppelt so hoch sein wie die maximale zu verarbeitende Frequenz, um keine Informationen zu verlieren. Die maximale Frequenz eines Signals, das verarbeitet werden soll, ist etwas höher als die Basisfrequenz des Channel 0. Der Channel 0 befindet sich im Frequenzbereich um die Basisfrequenz 1897,344 MHz. Nach der Verschiebung ins Basisband befindet sich dieser Channel also im Frequenzbereich um die Frequenz 1897,344 MHz − 1889,568 MHz = 7,776 MHz. Demnach wird mindestens eine Samplerate von 15 552 000 Samples/s benötigt. Um den vollständigen Channel 0 empfangen zu können, empfiehlt es sich, die Samplerate etwas zu erhöhen, um auch etwas höherfrequente Signale im Frequenzband des Channel 0 korrekt dekodieren zu können. Der HackRF One kann maximal 20 000 000 Samples/s verarbeiten, womit die minimal benötigte Samplerate überschritten wird. Als Samplerate werden die 20 000 000 Samples/s eingestellt.

Speicherung der Funk-Rohdaten und Offline-Signalverarbeitung

Jeder GNURadioCompanion-Flowgraph definiert den Datenfluss von einer oder mehreren *Sources* über verschiedene verarbeitende Blöcke zu einer oder mehreren *Sinks*. SDRs agieren als Signalquelle (*Source*). Nach durchgeführter Signalverarbeitung wird ein Signal an eine Ausgabe (*Sink*) gegeben und der GNURadioCompanion Flowgraph ist zu Ende.

Nativ unterstützt GNURadioCompanion einen *File Sink*, bei dem eingestellt werden kann, in welche Datei die Daten geschrieben werden sollen und welchen Typ das Eingangssignal (Complex, Int, Float, etc.) hat. Die Signalquelle *RTL-SDR Source*, die das Signal vom HackRF One liefert, kann einfach mit dem File Sink verbunden werden, um die Rohdaten des HackRF One zur späteren Verarbeitung zu speichern.

Bei diesem Vorgehen fallen pro Sekunde ca. 160 MB Daten an, die in Realzeit gespeichert werden müssen. Eigene Tests ergaben, dass klassische mechanische Festplatten mit ext4-Dateisystem eine Schreibgeschwindigkeit von ca. 100 MB bis 150 MB anbieten. Aus diesem Grund ist es lohnenswert, die Daten auf einem schnellen Speichermedium wie einer SSD zu speichern, um möglichen Datenverlust zu vermeiden und alle Samples speichern zu können.

Eine vom *File Sink* erzeugte Datei kann im Nachgang von der *File Source* gelesen werden und analog zu einer *RTL-SDR Source* als Signalquelle verwendet werden. Wenn in der Datei alle Samples gespeichert wurden, kann so offline und zeitunabhängig die gesamte Signalverarbeitung durchgeführt werden. Um nachträglich Zeitinformationen zu erhalten, kann die Nummer des aktuellen Samples ausgewertet werden, da die Samples mit einer konstanten Frequenz von 20 MHz abgetastet wurden und alle vorher gelesenen Samples der Reihe nach verarbeitet werden.

Demodulation des Signals der einzelnen DECT Channels

Der bestehende Flowgraph des gr-dect2 Projekts unterstützt lediglich die Verarbeitung eines einzelnen DECT Channels, der aus Sicht der verarbeitenden Blöcke in einem Basisband um die Frequenz 0 Hz liegt. Um die Offlinedaten korrekt auf die verschiedenen DECT-Channels aufzuteilen, müssen also die Signale der zehn DECT-Channels jeweils ins Basisband verschoben (*heruntergemischt*) und gefiltert werden.

Um die Basisfrequenz eines Signals zu ändern, muss ein sogenannter Mischer implementiert werden. Ein Mischer lässt sich mithilfe einer Multiplikation des Eingangssignals mit einem *lokalen Oszillator* implementieren. Hierzu wird das Eingangssignal mit der Frequenz f_I mit einem Kosinus-Signal multipliziert, das mit der Frequenz f_O schwingt, um die das Eingangssignal verändert werden soll. Nach dieser Multiplikation resultiert ein Signal mit der Basisfrequenz $f_I - f_O$. Wenn also beispielsweise der Channel 9 (das Frequenzband um 7,776 MHz) verarbeitet werden soll, muss ein Mischer mit einem lokalen Oszillator mit einer Frequenz von 7,776 MHz verwendet werden. Danach befindet sich das Signal von Channel 9 im Basisband um 0 Hz. Diese Form des Mischers muss für jeden Channel implementiert werden. Der Input wird deshalb mit zehn verschiedenen lokalen Oszillatoren multipliziert, sodass für jeden Channel ein separates Signal resultiert, dessen Frequenz 0 Hz der Basisfrequenz des Channels entspricht.

Nachdem sich das Signal eines Channels im Frequenzbereich $\left[\frac{-1{,}728}{2}\ \text{MHz};\ \frac{1{,}728}{2}\ \text{MHz}\right]$ befindet, müssen andere nicht zum Channel gehörende Signale (unter anderem die der restlichen Channels) herausgefiltert werden. Dies kann mithilfe eines Tiefpassfilters umgesetzt werden. Ein Tiefpassfilter verändert ein Signal derart, dass nur Signale mit einer Frequenz unterhalb einer einzustellenden Grenzfrequenz (f_{CO}) resultieren. Anteile des Eingangssignals mit höheren Frequenzen werden gedämpft. Ein Filter verarbeitet negative Signale analog zu ihrem positiven Pendant, sodass neben den Frequenzen $f > f_{CO}$ auch alle Frequenzen $f < -f_{CO}$ aus dem Eingangssignal herausgefiltert werden.

Zur Implementation eines Tiefpassfilters kann in GNURadio z. B. ein sogenannter FIR-Filter verwendet werden. Um diesen Filter zu erzeugen, müssen die sogenannten *Taps* berechnet werden. Sie leiten sich

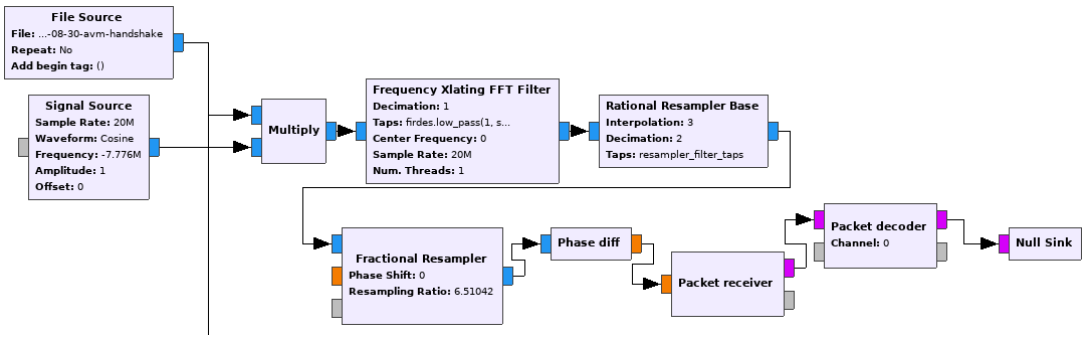


Abbildung 3.6: Verarbeitung des Signals von Channel 0

aus der Sample Rate, der Grenzfrequenz und einer Angabe zur Güte des Filters ab. Mithilfe von GNURadio lassen sich die Taps eines Tiefpassfilters mit der Funktion `firdes.low_pass(double gain, double sampling_freq, double cutoff_freq, double transition_width)`[2] erzeugen. Der Parameter `gain` gibt hier eine Vorverstärkung des Signals an, `sampling_freq` definiert die Samplerate des zu verarbeitenden Signals, bei `cutoff_freq` handelt es sich um die Grenzfrequenz und der Parameter `transition_width` ist eine Angabe zur Güte des resultierenden Filters (je kleiner der Wert, desto „steilflankiger“ der Filter, jedoch ist die Berechnung des Filters dann aufwändiger). Für die Filterung eines DECT-Channels können entsprechende Taps durch den Aufruf `firdes.low_pass(1, 20 000 000, 1 728 000/2, transition_width)` generiert werden, wobei `transition_width` während der Tests den willkürlichen Wert 1000 trug.

Die einzelnen Channels können nach der Filterung analog zu der ursprünglichen Version von gr-dect2 dekodiert werden (siehe Abbildung 3.6).

Anders als im ursprünglichen Flowgraph von gr-dect2 wird jedoch weder eine Audio-Dekodierung noch die Wiedergabe auf einem Sound-Device benötigt, weshalb die Daten, die der *Packet decoder* erzeugt, einfach an einen *Null Sink* weitergegeben werden können, um sie zu verwerfen.

Im folgenden Abschnitt wird beschrieben, wie die empfangenen Daten stattdessen gespeichert werden.

Versand des DECT-Payloads an ein Dummy-Interface

Der Block *Packet decoder* der gr-dect2 führt zunächst eine rudimentäre Verwaltung von Verbindungen durch, descrambled die B-Fields einer ausgewählten Verbindung und bereitet diese für die Audiowiedergabe vor. Anstatt einzelne Verbindungen zu filtern und die Daten dieser Verbindung zur Audiowiedergabe vorzubereiten, sollen im Rahmen dieser Arbeit jedoch abweichend alle Pakete als Rohdaten (also nicht descrambled) gespeichert werden. An dieser Stelle wird ein zum ReDECTed Projekt ähnliches Vorgehen implementiert, bei dem die unverarbeiteten DECT-Pakete in Ethernet-Frames verpackt und über einen Rawsocket an ein Dummy-Interface gesendet werden.

Neben dem D-Field des Pakets ist es zur späteren Analyse der Daten hilfreich, wenn möglichst viele Metadaten zum Paket vorliegen. Insbesondere wird zum Descrambling der Pakete die Framenummer benötigt. Aus diesem Grund ist es lohnenswert neben dem D-Field noch folgende weitere Werte in das Ethernet-Frame zu kodieren:

[2] *firdes* ist eine Kurzform für Finite Impulse Response (FIR) filter design functions

Rx Id, 4 Byte Eine interne eindeutige Id des Senders

Part Id, 5 Byte Sofern bereits erhalten der *Primary Access Rights Identifier* des Senders; eine Art Adresse, die innerhalb eines bestimmten A-Fields versendet wird

Framenumber, 1 Byte Die aktuelle Frame-Nummer

Timestamp, 8 Byte Die Samplenummer, an dem das Paket angekommen ist

Synchronisationswort, 3 Byte Das Synchronisationswort an dem das Paket begonnen hat

Channel, 1 Byte Der Channel, auf dem das Paket empfangen wurde

Grundlegend soll, um eine größtmögliche Kompatibilität zum bereits existierenden Wireshark-Dissector herzustellen, wieder der in Abbildung 3.2 definierte DECT-Header verwendet werden. In dieser Struktur ist jedoch kein Platz für die Werte *Rx Id* und *Part Id* vorgesehen, sodass diese außerhalb des DECT-Headers kodiert werden müssen. Hierzu bieten sich die sonst ungenutzten Felder *Source MAC Address* und *Destination MAC Address* des Ethernet-Headers an. Ferner muss zusätzlich der *Timestamp*-Wert kodiert werden können. Da in der bisherigen Header-Struktur kein weiterer Bereich umgedeutet werden kann, um diesen Wert zu speichern, muss der DECT-Header um 8 byte erweitert werden, sodass die in Abbildung 3.7 visualisierte Struktur resultiert.

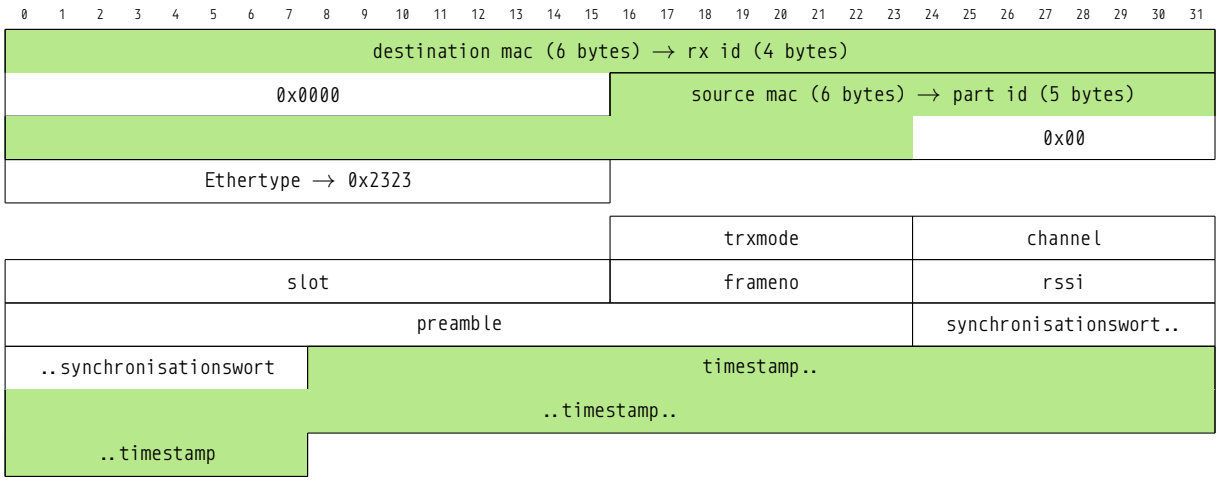


Abbildung 3.7: Angepasster DECT-Header

Anders als im ReDECTed Projekt werden alle Felder des DECT-Headers nun korrekt befüllt (bis auf das nicht genutzte `preamble`-Feld).

Da ULE bei bestimmten Paketen, den sogenannten *ULE Dummy Bearer*-Paketen, kein Scrambling durchführt, wird der Payload nicht bereits im *Packet decoder*-Block descrambled, sondern in Originalform an den Rawsocket übergeben. Aus diesem Grund kann die Erzeugung und der Versand des Ethernet-Frames im Quelltext des *Packet decoders* direkt zu Beginn er `work()`-Funktion durchgeführt werden. Da der *Packet decoder* jedoch bereits einige Metadaten zu den Verbindungen speichert (z. B. den Wert `part_id`) ist es lohnenswert, zunächst die Verbindungsverwaltung des *Packet decoder* durchzuführen und die darin ermittelten Daten auch im DECT-Header zu kodieren.

Der zum Versand benötigte Rawsocket wird direkt beim Erzeugen der Instanz der Klasse in ihrem Konstruktor erstellt. In der aktuellen Implementierung wird standardmäßig das Interface mit dem Namen

`dummy0` als Ziel ausgewählt, sodass vor der Instanziierung eines Objektes dieser Klasse zunächst ein passendes Netzwerkinterface erstellt und konfiguriert werden muss. In aktuellen Linux-Distributionen lässt sich ein passendes Dummy-Interface mithilfe des Werkzeugs `ip` wie folgt erzeugen und aktivieren:

```
ip link add dummy0 type dummy
ip link set dummy0 up
```

Abbildung 3.8: Einrichtung eines Dummy-Netzwerkinterface mit dem `ip`-Tool

Durch die Nutzung eines Rawsockets ergibt sich jedoch eine Limitierung: die Demodulation muss mit Root-Rechten durchgeführt werden, da nur Superuser unter Linux Rawsockets nutzen dürfen.

3.2.4 ULE Parsing

Die in Wireshark vorliegenden Daten sollen im nächsten Schritt analysiert werden. Hierzu werden die als `pcap`-Datei vorliegenden Daten geparst. An ein Analysewerkzeug für ULE Traffic werden insbesondere folgende funktionale Anforderungen gestellt:

Parsing der Nachrichten Die empfangenen Pakete sollten auf allen definierten Layern geparst werden. Die Applikationsdaten sollten als Bytestrom zur weiteren manuellen Analyse ausgegeben werden können.

Durchsuchbarkeit der Daten Die empfangenen Pakete sollten durchsuchbar sein. Insbesondere sollten einzelne Verbindungen ausgewählt werden können, um diese separat analysieren zu können.

Erweiterbarkeit um weitere Nachrichtentypen Für etwaige Erweiterungen des Protokolls sollten, ohne die Architektur der Software anzupassen, weitere Nachrichtentypen implementiert werden können.

Verwaltung von Bearern & Verbindungen Pakete, die zu einer Verbindung zwischen zwei Kommunikationspartnern gehören, sollten zusammen betrachtet werden können.

Hohe Performance Um auch Online-Attacken gegen ULE unterstützen zu können, muss die Software Echtzeitanforderungen genügen können.

Da der von Wireshark implementierte Parser nur ein sehr rudimentäres Parsing der Pakete durchführt und auch eine Erweiterung dieses Parsers um die oben genannten Punkte, insbesondere in der Programmiersprache C, einen sehr hohen Aufwand darstellen und sehr fehleranfällig sein würde, wird ein neuer Parser `rule` (Akronym für *Rust ULE*) für DECT bzw. ULE in der Programmiersprache Rust entwickelt. Mit diesem Parser sollen ULE Pakete analysiert werden können. Mit einer umfangreichen ULE Implementierung lassen sich mit der Software zunächst offline Angriffe auf das ULE Protokoll durchführen, wie beispielsweise die Berechnung des *UAK* auf Basis eines aufgenommenen *Easy Pairing*-Szenarios. Grundsätzlich könnten auch verschiedene Datenquellen konfiguriert werden, um in verbesserten Versionen der Toolchain (insbesondere mit einer verbesserten Implementation der Signalverarbeitung) auch Online-Angriffe durchführen zu können.

Im Folgenden wird die Implementation dieser Software genauer beschrieben.

Architektur

Um in der Implementation von `rule` einen starken Bezug zur DECT und ULE Spezifikationen zu behalten, werden analog zu den Protokollschichten im DECT Protokollstack vier Module mit den Namen `phl`, `mac`, `dlc` und `nwk` definiert. Zudem wird ein Modul `dump` implementiert, das Dateien im `pcap`-Format verarbeitet und alle darin enthaltenen DECT-Pakete nacheinander an die Implementation des PHL-Layers übergibt.

Jedes in `rule` definierte Modul implementiert die Funktionen, die es laut der jeweiligen Spezifikation zu übernehmen hat. Eine Ausnahme ist hier das `phl`-Modul, da die eigentliche Funktionalität dieses Moduls bereits in GNURadio implementiert ist.

Das `MAC`-Modul implementiert die Verwaltung der Bearer, das Parsing des A-Field und das Parsing des B-Field – sofern es zur Übertragung von MAC-Nachrichten genutzt wird. Das `dlc`-Modul implementiert das Reassembling der *U-Plane-Messages*, also der Applikationsdaten.

Unter anderem für das Reassembling von fragmentierten Nachrichten ist es notwendig, Metadaten wie Verbindungsinformationen und Puffer für Daten paketunabhängig in einem Status speichern zu können. Hierzu wird die Struktur `UleContext` definiert, in der für jede identifizierte MAC-Verbindung Daten innerhalb eines Elements der Struktur `Connection` gespeichert werden können. Insbesondere werden hier die Werte `request_no`, `confirm_no` und `release_no` gespeichert, die jeweils die Samplenummer des Pakets angeben, in dem die Verbindung angefragt, aufgenommen und beendet wurde. Neben diesen Werten wird hier unter anderem auch der Puffer für das Reassembling des DLC-Layers gespeichert.

In jedem ULE-(DECT-)Layer gibt es einige verschiedene Nachrichtentypen. So gibt es auf dem MAC-Layer im A-Field beispielsweise `IdentitiesInformation`-Nachrichten, die ein Teilnehmer nutzt, um seinen Identifier mitzuteilen, oder die `SystemInformation`-Nachrichten, die der FP unter anderem dazu nutzt, um die TDMA-Struktur aller Teilnehmer zu synchronisieren; der DLC-Layer definiert unter Anderem `FU10a`, `FU10b` und `FU10c`-Nachrichten. Diese verschiedenen Nachrichtentypen werden pro Layer zusammengefasst. In jedem zu einem Layer gehörenden `rule`-Modul wird dazu ein `enum` definiert, das alle auf dem Layer implementierten Nachrichtentypen enthält. Durch diese Struktur ist ein fester Ablauf definiert, wie weitere Nachrichtentypen in das Projekt einzupflegen sind. Hierzu sind lediglich zwei Anpassungen erforderlich: Zunächst muss ein entsprechender weiterer Eintrag in das jeweilige `enum` hinzugefügt werden, um festzulegen, dass auf dem entsprechenden Layer eine neue Nachricht definiert ist. Im zweiten Schritt muss die für das `enum` definierte `parse()`-Funktion um die Funktionalität erweitert werden, den hinzuzufügenden Nachrichtentyp zu erkennen und abhängig davon die für den Nachrichtentyp definierte `parse()`-Funktion des Nachrichtentyps aufzurufen.

Durch die Verwendung von `enum`-Typen für die Nachrichtentypen der jeweiligen Netzwerk-Layer ist es sehr einfach den Parser um weitere Nachrichtentypen zu erweitern. Auch das Hinzufügen weiter Layer ist unkompliziert: Es wird ein neues Modul erzeugt, das wiederum ein `enum` für alle Nachrichtentypen besitzt und eine `parse()`-Funktion implementiert. Diese Funktion muss dann nur an der geeigneten Stelle im übergeordneten Layer aufgerufen werden.

Parsing

Wenn ein Paket geparst werden soll, muss eine Instanz der `Packet`-Struktur erzeugt werden. Wenn auf dieser Instanz die `parse()`-Methode aufgerufen wird, wird das gesamte Paket auf allen Layern geparst

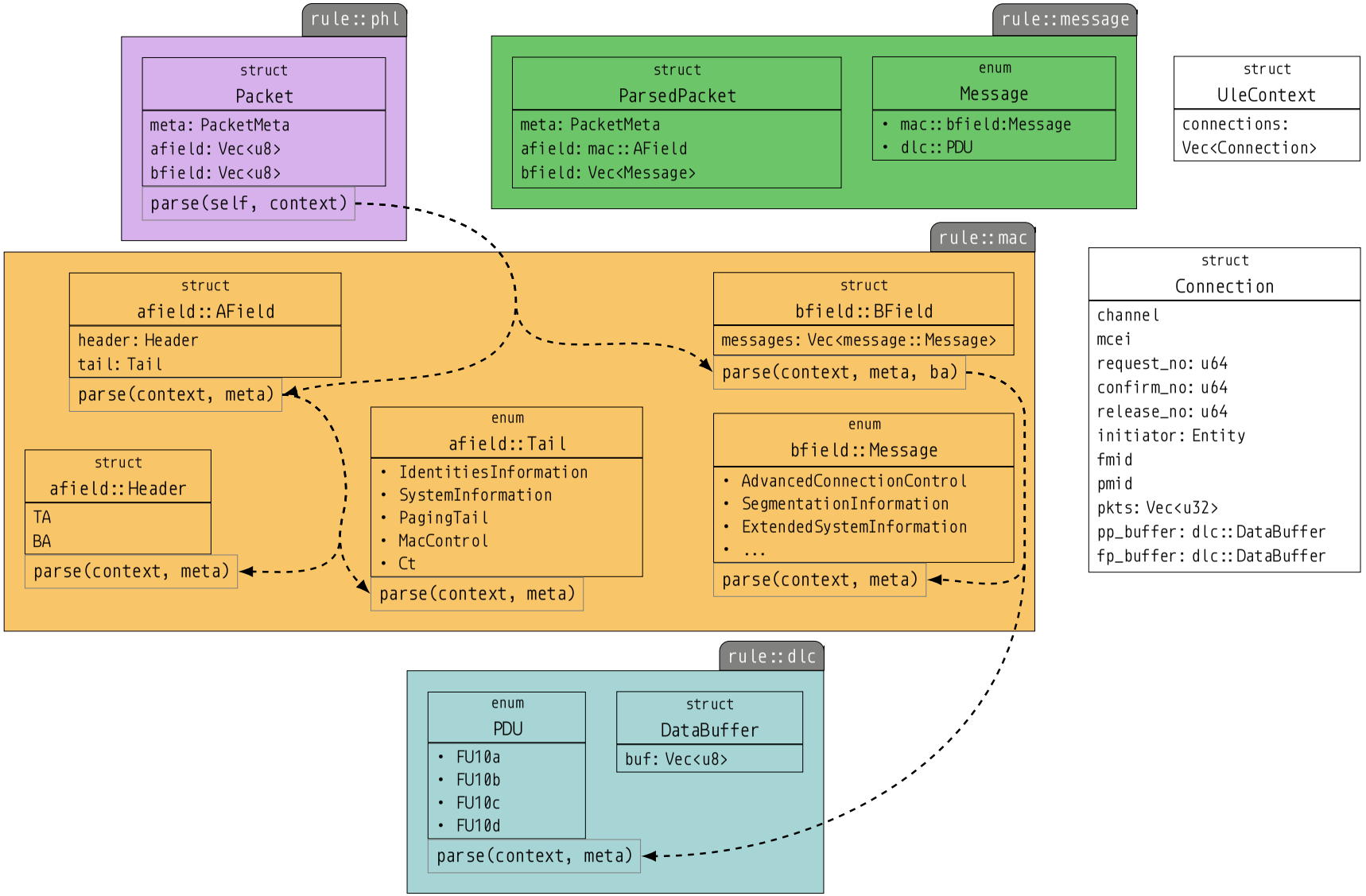


Abbildung 3.9: Architektur des `rule`-Parsers mit grundlegendem Kontrollfluss

(siehe gestrichelten Kontrollfluss in Abbildung 3.9). Jede Schicht und jeder Nachrichtentyp innerhalb einer Schicht besitzt eine eigene statische `parse()`-Funktion, die ein Byte-Array entgegennimmt, parst und daraus eine Instanz des entsprechenden Typs erzeugt. Diese Funktion kann ihrerseits wieder `parse()` -Funktionen anderer Typen aufrufen.

Beim Parsing werden innerhalb des `phl::Packet`-Typs zunächst die Rohdaten der in der `pcap`-Datei vorhandenen Pakete verarbeitet. Der Ethernet-Header und der DECT-Metadaten-Header werden geparst und vom eigentlichen DECT-Payload getrennt. Des Weiteren implementiert das `phl`-Modul Prüfsummen-Checks, mit deren Hilfe überprüft werden kann, ob das A-Field und das B-Field korrekt empfangen wurden.

Im `mac`-Modul werden diverse Funktionen des MAC-Layers implementiert. Das Modul ist primär in zwei Untermodule unterteilt: `afield` und `bfield`. Das A-Field eines Paketes beinhaltet zunächst im Header (`afield::Header`) zwei wichtige Werte: `TA` und `BA`. Bei beiden Werten handelt es sich um eine Art Messagetype-Angabe für nachfolgende Daten: `TA` definiert, welche Daten im sogenannten *Tail* des A-Fields (die letzten 5 byte des A-Field) vorliegen und `BA` definiert, welche Daten im B-Field des Pakets vorliegen und wie diese zu verarbeiten sind.

Um ein B-Field (`mac::BField`) korrekt analysieren zu können, müssen zwei Informationen vorliegen: Zunächst muss definiert sein, um was für eine Art B-Field es sich handelt: Werden Applikationsdaten oder Kontrolldaten übermittelt, oder handelt es sich um den sogenannten ULE Dummy Bearer? Um dies zu ermitteln, wird der im A-Field kodierte Wert `BA` ausgewertet. Handelt es sich um ein reguläres B-Field (*kein* ULE Dummy Bearer), so muss es descrambled werden. Welche Bitmaske zum Descrambling des Pakets verwendet werden muss, ist abhängig von der Nummer des Frames, in der das Paket versendet wurde. Es gibt 16 verschiedene Bitmasken die sich in jedem Multiframe wiederholen.

Ein descrambletes B-Field wird anschließend – je nach Format des B-Fields, welches auch im `BA`-Feld kodiert ist, – geparst. Neben dem ULE Dummy Bearer kann das B-Field in einem von drei Formaten kodiert sein: es können entweder mehrere MAC-Layer Nachrichten(im sogenannten *E-Type*-Modus), eine MAC-Layer Nachricht und eine oder mehrere U-Plane Nachrichten (*E+U-Mux*-Modus) oder ausschließlich U-Plane Daten (*U-Type*-Modus) im B-Field kodiert sein.

Wenn im B-Field U-Plane Daten vorliegen, werden diese an den DLC-Layer weitergegeben (`dlc::PDU::parse()`), ansonsten werden die Nachrichten im MAC-Layer verarbeitet.

Beim Parsing einer U-Plane PDU (Protocol Data Unit) wird der Payload zunächst an den entsprechenden `dlc::DataBuffer` angehangen. Für jede Verbindung wird für beide Verbindungsrichtungen je ein `dlc::DataBuffer` gespeichert. Nachdem die empfangenen Daten angehängt wurden, wird geprüft, ob bereits ein oder mehrere vollständige SDUs (Service Data Unit) empfangen wurden. Wenn dies der Fall ist, werden die Rohdaten dieser SDUs an den NWK-Layer zur weiteren Analyse übergeben.

Das eigentliche Parsing ist auf allen Layern mithilfe des Parser-Combinator-Frameworks `nom`[3] implementiert. Diese Bibliothek lässt ein Parsing der Nachrichten auf Bit-Ebene zu und ist so zur Analyse von Netzwerkprotokollen sehr gut geeignet.

[3] siehe `https://github.com/Geal/nom`

Verwaltung von Bearern & Verbindungen

Die belegten Timeslots und somit die verwendeten Bearer werden vom MAC-Layer verwaltet und als Verbindungen zusammengefasst. Aus diesem Grund ist die Verbindungsverwaltung im `mac`-Modul implementiert. Sobald eine MAC-Nachricht vom Typ `AccessRequest` empfangen wird, wird eine neue Instanz der `Connection` Struktur erzeugt und im `UleContext` gespeichert. In dieser Connection werden zunächst nur die Felder `request_no` (befüllt mit der aktuellen Samplenummer), `channel` (befüllt mit dem DECT-Channel, auf dem das Paket empfangen wurde) und `initiator` (befüllt mit dem Wert PP bzw. FP) belegt und das aktuelle Paket wird zur in der `Connection` gespeicherten Paketliste `pkts` hinzugefügt.

Wenn nun weitere Nachrichten empfangen werden, wird immer geprüft, ob eine Nachricht zu einer bestehenden Verbindung gehört. Da die Zuordnung von Nachrichten zu Verbindungen ausschließlich über die Zuordnung des entsprechenden Timeslots zur Verbindung möglich ist, muss auch in `rule` eine Möglichkeit geschaffen werden, das exakte Timing der Nachricht zu bestimmen. Nachrichten, die zum selben Bearer gehören, müssen immer entweder exakt eine halbe Frame-Dauer auseinander liegen (Duplex-Bearer mit exakt zwei Timeslots) oder exakt eine Frame-Dauer (Simplex-Bearer).

Da die Samples mit einer konstanten Rate erzeugt wurden, lässt sich aus den Samplenummern auf die Zeit des Empfangs des Pakets schließen. Um die Länge eines Frames in Sample-Zahlen zu berechnen, bietet es sich an, das `frameno`-Feld in Zusammenhang zum `timestamp`-Feld aus dem DECT-Header (siehe Abbildung 3.2) auszuwerten. Durch das umfangreiche Resampling der Signale in den *Rational Resampler Base* und *Fractional Resampler*-Blöcken in GNURadioCompanion kann jedoch nicht davon ausgegangen werden, dass in einer Sekunde exakt 20 000 000 Samples im Dump gezählt werden und so die Framedauer 2 000 000 Samples beträgt.

Zur Berechnung der realen Frame-Dauer in Samples wird stattdessen folgender Sachverhalt angenommen: Das erste Paket, das in Frame i ankommt, kommt in der Regel genau eine Frame-Dauer vor dem ersten Paket des Frames $i + 1$ an. Die Differenz der Samplenummern der jeweils ersten Pakete zweier aufeinanderfolgender Frames entspricht dann der Dauer eines Frames in Samples. Diese Annahme ist in Abbildung 3.10 visualisiert.

Bei Funkkommunikation können immer einzelne Pakete verloren gehen. Aus diesem Grund ist die Berechnung der Frame-Dauer über die beschriebene Methodik nicht immer korrekt: Wenn das erste Paket in einem Frame nicht korrekt empfangen wird, wird die Frame-Dauer des vorangehenden Frames falsch berechnet, sodass eine längere Frame-Dauer berechnet wird, als real zutrifft. Auch die Länge des folgenden Frames wird dann falsch gemessen und es resultiert ein zu kleiner Wert für die Frame-Dauer. In Abbildung 3.11 ist eine Zahlenreihe von gemessenen Framelängen dargestellt, die nach gemessener Länge sortiert ist. Die y-Achse der Grafik entspricht der berechneten Framelänge und die x-Achse bezieht sich auf den Index in der sortierten Zahlenreihe.

Um die korrekte Framelänge zu berechnen, wird davon ausgegangen, dass mindestens $\frac{1}{3}$ der berechneten Paketlängen korrekt ist, sodass sich *maximal* in den $\frac{1}{3}$ kleinsten Messwerten und *maximal* den $\frac{1}{3}$ größten Messwerte Ausreißer befinden. In der Grafik ist zu erkennen, dass diese Annahme bei den dort verwendeten Messwerten zutrifft. Der Mittelwert des übrigen Drittels der berechneten Framelängen wird als Referenz-Framelänge für die Verbindungserkennung verwendet. Hierbei wird eine Toleranz von $\frac{1}{48}$ der errechneten Framedauer gewährt. Das entspricht der Zeit eines halben Slots, sodass bestmöglich ausgeschlossen wird, dass ein Slot falsch zugeordnet wird und gleichzeitig eine größtmögliche Varianz in im Timing zugelassen wird.

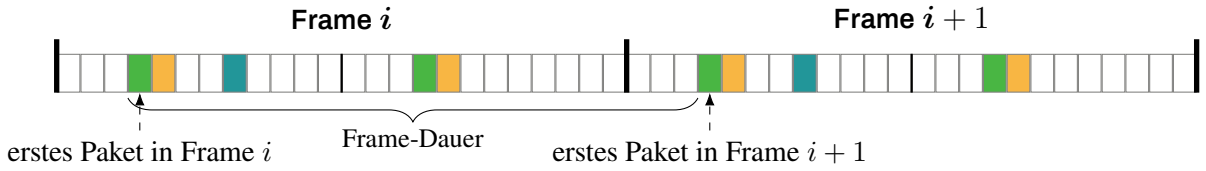


Abbildung 3.10: Berechnung der Frame-Dauer in Samples

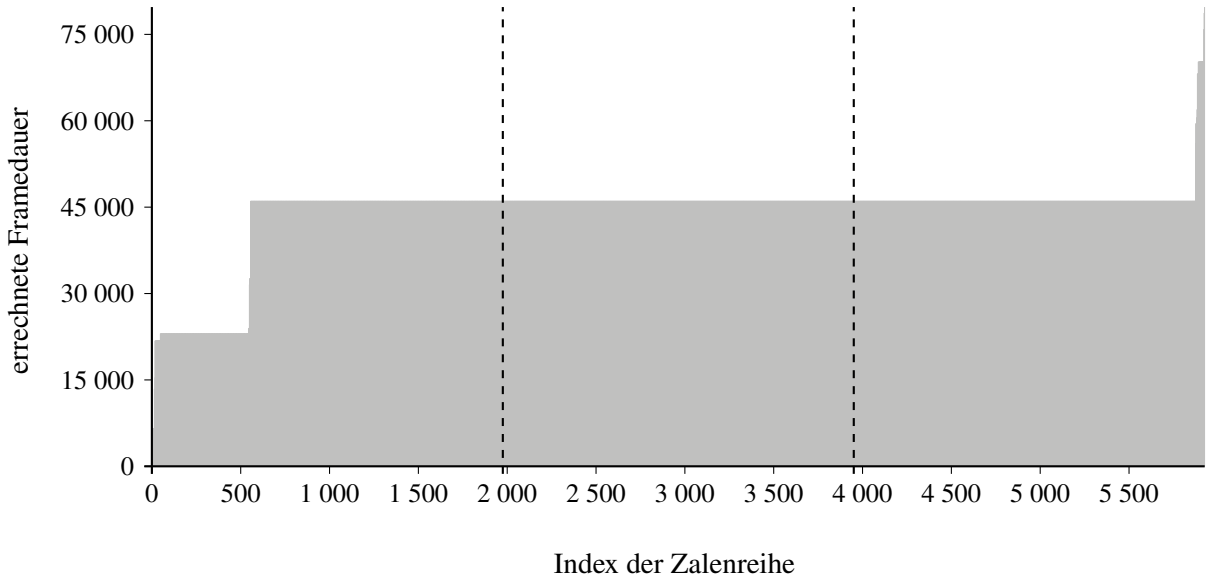


Abbildung 3.11: Gemessene Frame-Dauer mit eingetragenen Grenzen für die $\frac{1}{3}$ kürzesten und $\frac{1}{3}$ längsten Messergebnissen

Filterung der Pakete

Nach dem Parsing können die Pakete in einer Liste gespeichert werden, welche dann im Anschluss durchsucht werden könnte. Da dieser Ansatz jedoch sehr statisch ist und in jedem Fall die Anwendung angepasst und neu kompiliert werden müsste, ist es lohnenswert eine dynamischere Möglichkeit für diesen Use-case zu implementieren. Eine Möglichkeit hierzu ist die Implementation einer Schnittstelle zu einer einfachen Skriptsprache, die zur Laufzeit von `rule` ausgewertet wird. Eine in Rust einfach einzubettende Sprache, die dazu geeignet ist, diese Schnittstelle bereitzustellen ist `dyon`. In `dyon` implementierte Funktionen lassen sich problemlos aus Rust heraus aufrufen und in `dyon` zu verwendende Strukturen können einfach in Rust definiert werden. Für einen Anwender ist die Sprache recht einfach zu verwenden, da die benötigte Laufzeitumgebung direkt in `rule` mitgeliefert wird.

Um einen `dyon`-Filter für `rule` zu definieren und so nur bestimmte Pakete auszugeben, kann die Kommandozeilenoption `--filter FILTER` genutzt werden. Wenn ein Filter angegeben wird, beeinflusst dies nur die Ausgabe der Pakete; analysiert werden in jedem Fall alle Pakete, da beispielsweise die Verbindungsverwaltung sonst nicht möglich wäre. Bei dem Filter handelt es sich um einen Bool'schen-Ausdruck, der mit einem verarbeiteten Paket aufgerufen wird. Nur wenn der Bool'sche Ausdruck wahr ist, wird das Paket ausgegeben, ansonsten nicht.

```
rule print \
    handshake.pcap \
    --filter '(p.no > 6737) && (p.channel == 7) \
        (p.parsed_afield?.bmux != "NoBfield")'
```

Abbildung 3.12: Aufruf des `rule`-Dissectors mit Filter

3.2.5 Zusammenfassung

Zusammenfassend lässt sich der Datenfluss des Mitlesens der Funkkommunikation in drei Schritte gliedern:

1. Die Funk-Rohdaten werden gelesen und abgespeichert.

   Mithilfe von GNURadio werden die Funk-Rohdaten auf eine SSD geschrieben.

2. Die Signale werden analysiert, gefiltert und demoduliert und das Resultat wird als `pcap` abgespeichert.

   Durch eine Anpassung des gr-dect2-Projekts (siehe [gr-dect2]) werden die Daten nun an ein Dummy-Netzwerkinterface gesendet. Unter Zuhilfenahme von Wireshark können die Rohdaten der Pakete in einer `pcap`-Datei zur weiteren Analyse gespeichert werden.

3. Die Daten werden geparst.

   Zum Parsing der Daten wurde eine Rust-Anwendung implementiert. Sie kann einen `pcap`-Dump verarbeiten und alle darin enthaltenen Pakete parsen, Verbindungen verfolgen und ist dahingehend erweiterbar, dass weitere Nachrichtentypen und Netzwerk-Layer hinzugefügt werden können.

3.3 Ergebnisse & Ausblick

Zum Test der Implementierung und als reale Fallbeispiele für DECT ULE Geräte wurden zwei Systeme genutzt: eine Alarmanlage von Panasonic und eine AVM FRITZ!Box mit einem AVM FRITZ!DECT 301 Heizungsthermostat (siehe Abschnitt 3.1 „Testumgebung“). Bei beiden System fällt auf, dass zum Pairing keine Werte Out-of-Band ausgetauscht werden müssen, sondern lediglich ein Knopfdruck zur Initiierung des Pairings ausreicht. Dies lässt darauf schließen, dass in der Tat entweder das *Easy Pairing*-Szenario (siehe Abschnitt 2.7.6 „Easy Pairing“) oder ein vordefinierter statischer *AC* zum Schlüsselaustausch verwendet wird.

Die Kommunikation von einem PP zum FP ließ sich bei beiden Systemen mithilfe der angepassten gr-dect2-Implementierung und dem `rule`-Parser mitlesen. Auch das Verfolgen einzelner PP/FP Verbindungen ist hiermit möglich, obwohl die Signalverarbeitung und das Parsing im aktuellen Stand keinen Echtzeitanforderungen genügen. Darüber hinaus ist im Parser die Analyse von Nachrichten auf dem MAC-Layer bereits korrekt implementiert, was zum einen für die Verwaltung von Verbindungen notwendig ist und zum anderen allerdings auch den Inhalt von im A-Fields versendeten Nachrichten offenlegt. Hier kodierte Daten sind in aller Regel nicht verschlüsselt. Eine erste Implementation des DLC- und NWK-Layers ist auch erstellt, sodass auch ein Reassembling-Mechanismus – wie im DLC-Service *LU10* definiert (siehe Abschnitt 2.4.4 „Data Link Control (DLC)“) – innerhalb des `DataBuffer` implementiert ist.

Aufgrund der unvollständigen Implementierung des DLC- und NWK-Layers ist es im Rahmen dieses Projekts noch nicht gelungen, das Pairing zweier Geräte vollständig zu analysieren. Der erstellte Parser kann jedoch dahingehend erweitert werden, dass ein vollständiges Dissecting von Nachrichten auf diesen Layern implementiert wird und damit die ausgetauschten Challenge und Response Werte mitgelesen werden könne. Aufbauend auf diesen Werten lässt sich dann der A Algorithm berechnen (im Falle von *Easy Pairing*) bzw. brechen (im Falle einer regulären *Key Allocation* mit kurzem AC) und die weitere Kommunikation einer Verbindung entschlüsseln.

Um auch online Angriffe gegen das ULE Protokoll durchzuführen, müsste zunächst Echtzeitfähigkeit in den einzelnen Softwaremodulen hergestellt werden. Wenn die Signalverarbeitung hinreichend performant arbeitet, kann der `rule`-Parser die Daten auch direkt vom Dummy-Netzwerkinterface empfangen und ad-hoc verarbeiten. Eine weitere Einschränkung, die es mit dem in dieser Arbeit beschriebenen Setup verhindert aktive Angriffe durchzuführen, ist, dass der HackRF One nur über einen halb-duplex Transceiver verfügt. Somit kann der HackRF One nicht während des Lesens der Funkkommunikation gezielt Nachrichten injizieren oder die Übermittlung einzelner Nachrichten zu stören. Als Alternative zum HackRF One könnte hier beispielsweise das *LimeSDR*[4] verwendet werden, das einen full duplex Transceiver besitzt. Wenn ein SDR verwendet wird, das auch in der Lage ist, Nachrichten zu versenden muss zusätzlich Funktionalität entwickelt werden, die es erlaubt ULE-Nachrichten zu kodieren und diese an GNURadio zu übergeben.

Um die Performance der Signalverarbeitung zu verbessern, könnte beispielsweise das Filterdesign optimiert werden. Sowohl der Typ der verwendeten Tiefpassfilter (*Frequency Xlating FFT Filter*) als auch deren Parametrisierung sind für ihren Einsatzzweck überdimensioniert, was einen zu hohen Rechenaufwand bedingt. Mit den in Abschnitt 3.2.3 „Demodulation des Signals der einzelnen DECT Channels“ festgelegten Parametern werden für jeden Filter 48181 Taps definiert. Im Vergleich zum ReDECTed Projekt, das mithilfe von lediglich 95 Taps einen Filter innerhalb eines *Polyphase Channelizer* implementiert,

[4] siehe `https://limemicro.com/products/boards/limesdr/` und `https://github.com/myriadrf/LMS7002M-docs`

mit dem alle zehn DECT-Channels dekodiert werden können, ist offensichtlich, dass hier enormes Optimierungspotential besteht.

Wenn die Performance der Gesamtlösung optimiert ist und die Implementation des DLC- und NWK-Layer vervollständigt ist, ist ein aktiver Angriff gegen das Protokoll mit der hier beschriebenen Methodik – insbesondere ohne spezielle DECT-Hardware wie die vom DeDECTed-Projekt verwendete *Dosch + Amand COM-ON-AIR*-Karte – ohne größeren Aufwand möglich. Lediglich ein SDR mit einer Bandbreite von 20 MHz, einer Samplerate von 20 000 000 Samples/s und einem full-duplex Transceiver wäre hierzu notwendig.

4 Fazit

Viele Smart Home Systeme nutzen Funk als Kommunikationskanal, um die Hürde bei einer Nachrüstung in Häusern so gering wie möglich zu halten. Einige Einsatzszenarien von Smart Home Geräten – inbesondere bei der Gebäudeautomatisierung oder in Sicherheitsanwendungen wie Alarmanlagen – sind jedoch beliebte Ziele für Angreifer.

Anders als bei kabelgebundener Kommunikation ist der Zugriff auf die Funkkommunikation nicht durch physische Barrieren wie Wände und verschlossene Türen gegen unautorisierten Zugriff geschützt. Ein Angreifer muss nicht zunächst Zugang zu einem Kabel erlangen, um mit Geräten im funkbasierten Smart Home Netzwerk zu kommunizieren. Aus diesem Grund ist eine Sicherheitsbetrachtung von Funkprotokollen im Smart Home Bereich sehr wichtig.

Das Smart Home Protokoll ULE ist als Profil des in den 90er Jahren entstandenen Funkprotokolls DECT definiert. Bei DECT handelt es sich um ein Funkprotokoll, das die Sterntopologie nutzt. Zentral ist in jedem DECT-Netzwerk die *Basisstation*, welche die gesamte Kommunikation koordiniert und abwickelt. Neben der Basisstation existieren *Mobilteile*, die sich mit der Basisstation verbinden.

Schon in frühen Versionen des DECT Protokolls wurden Algorithmen zur Authentifizierung von Teilnehmern und zur Verschlüsselung von Nachrichten festgelegt. Die ersten Algorithmen, die hierzu verwendet wurden, *DSAA* zur Authentifizierung und *DSC* zur Verschlüsselung, sind bis heute nicht offiziell veröffentlicht, jedoch vom deDECTed Projekt umfangreich analysiert und zum Teil gebrochen worden (vgl. [DedectedDSAA] und [DedectedDSC]). Aus demselben Projekt sind neue, auf AES-128 basierende, Algorithmen (DSAA2 und DSC2) entstanden, welche als Alternative zu den *alten* Algorithmen verwendet werden können.

Die neuesten Spezifikationen des DECT Standards unterstützen auch die Verschlüsselung der Applikationsdaten auf Basis von CCM (siehe [RFC 3610]) mit AES-128 als Verschlüsselungsprimitive. Dabei werden Nutzdaten zusätzlich mit einem Message Authentication Code ausgestattet, den ein Angreifer nicht fälschen kann. Er kann folglich keine Nachrichten manipulieren oder injizieren, ohne den symmetrischen AES-Schlüssel zu kennen.

Immer, wenn eine neue Verbindung zwischen der Basisstation und einem Sensor bzw. Aktor aufgebaut wird, authentifizieren sich die Geräte gegenseitig. Bei dieser Authentifizierung wird während eines Challenge-Response-Verfahrens, das durch den Authentifizierungsalgorithmus DSAA/DSAA2 definiert ist, ein neuer Schlüssel ausgetauscht. Dieser Schlüssel wird zur Verschlüsselung mit AES-CCM der nachfolgenden Kommunikation genutzt.

Wenn beispielsweise ein Fenstersensor detektiert, dass das Fenster geöffnet wird und dies der Basisstation mitteilen möchte, baut der Fenstersensor zunächst eine Verbindung zu dieser auf. Zu Beginn dieser Verbindung authentifizieren sich beide Partner gegenseitig. Hierbei initiiert die Basisstation das *Authentifizierung eines PP* Szenario. Während dieses Szenarios startet der Fenstersensor das *Authentifizierung eines FP*, um eine gegenseitige Authentifizierung zu erreichen.

Mit dem erfolgreichen Abschluss des *Authentifizierung eines PP* Szenarios haben beide Partner ein gemeinsames Geheimnis, den *Derived Cipher Key*. Dieser Schlüssel wird in der folgenden Kommunikation als Schlüssel für die Verschlüsselung mit AES-CCM genutzt. Die für CCM benötigte Nonce wird aus einer aufsteigenden Sequenznummer und mehreren wohldefinierten Verbindungsmetadaten zusammengesetzt. Somit kann ausgeschlossen werden, dass eine Nonce mehrfach verwendet wird.

Ein Angreifer kann sich nicht als legitimer Netzwerkteilnehmer ausgeben. Um die Authentifizierung durchzuführen, muss er den geheimen Wert des des 128 bit langen Authentication Key UAK kennen. Diesen Wert zu brechen ist nach aktuellem Wissensstand praktisch unmöglich.

Zur Verschlüsselung mit AES-CCM wird ein während der *Authentifizierung des PP* abgeleiteter *Derived Cipher Key* verwendet. Da dieser Schlüssel aus dem ihm unbekannten UAK abgeleitet wird, kann ein Angreifer augetauschte Nachrichten weder entschlüsseln noch manipulieren, da er nicht in der Lage ist, den Message Authentication Code der Nachrichten korrekt neu zu berechnen. Gespoofte oder manipulierte Nachrichten werden vom Empfänger direkt verworfen.

Wenn ein Angreifer jedoch lediglich die Kommunikation eines Fenstersensors mit der Basisstation unterbinden möchte, genügt es, die DECT Funkverbindung zu stören. In diesem Fall ist der Fenstersensor zwar in der Lage, die fehlenden Antworten der Basisstation zu detektieren – da es jedoch keine alternative Schnittstelle des Fenstersensors gibt, über die er über ein geöffnetes Fenster informieren kann, kann das Ausbleiben der Nachrichten von einer Basisstation nicht erkannt werden.

4.1 Schwachstellen

Im Rahmen der in dieser Arbeit durchgeführten Sicherheitsanalyse des ULE Protokolls sind neben unmittelbar sicherheitsrelevanten Problemen auch weitere potentiell sicherheitskritische Eigenschaften des Protokolls erkannt worden. Die identifizierten Schwachstellen von ULE werden im Folgenden zusammengefasst und erläutert.

4.1.1 Unsicheres Pairing

Die Überprüfung der Authentizität eines neuen Netzwerkteilnehmers wird bei DECT durch den Nutzer durchgeführt. Dieser tauscht den sogenannten Authentication Code (AC) als initiales Geheimnis manuell (also Out-of-Band) aus. Aufbauend auf diesem kurzen Wert, der in der Regel bis zu 32 bit lang ist, berechnen die Geräte (die Basisstation und das neue Mobilteil) im Protokollszenario *Key Allocation* einen neuen starken 128 bit Schlüssel. Dieser Schlüssel wird User Authentication Key (UAK) genannt. Der UAK wird bei der Authentifizierung als Authentication Key genutzt und damit auch dazu verwendet, neue Geheimnisse zur Verschlüsselung der einzelnen Verbindungen abzuleiten.

Wenn ein Angreifer die Kommunikation während des *Key Allocation* Szenarios aufzeichnet, ist er im Nachhinein offline in der Lage, den verwendeten Schlüssel zu brechen. Er kann mithilfe von Brute-Force den genutzten 32 bit Schlüssel erraten. Hierzu muss er lediglich mit allen möglichen AC-Werten das Challenge-Response Verfahren berechnen. Sobald eine vom Angreifer berechnete Response mit der mitgelesenen Response übereinstimmt, hat er den korrekten AC gefunden. Im Mittel muss der Challenge-Response Algorithmus also lediglich 2^{31} mal berechnet werden, um den korrekten AC zu erraten. Häufig werden sogar nur vier Ziffern als AC verwendet, sodass sich der Aufwand auf bis zu 10 000 Versuche

limitiert. Wenn der AC bekannt ist, kann ein Angreifer den abgeleiteten UAK analog zu den legitimen Kommunikationspartnern berechnen.

Für ULE ist eine noch weitaus weniger sichere Variante des Pairings definiert: das *Easy Pairing*. Hierbei wird derselbe Algorithmus wie bei der *Key Allocation* durchgeführt. Anders als bei der *klassischen Key Allocation* verfolgt das *Easy Paring* jedoch einen Trust-On-First-Use-Ansatz. Es wird zuvor kein Geheimnis Out-of-Band ausgetauscht, sondern der konstante Wert 0000 als AC festgelegt.

Wenn ein Angreifer die Kommunikation dieses Szenarios mitliest, kann er, sogar ohne einen Schlüssel brechen zu müssen, den resultierenden UAK berechnen, indem er einfach die gleichen Algorithmen durchführt, wie die beteiligten Kommunikationspartner.

Wenn ein Angreifer den UAK zweier Kommunikationspartner kennt, kann er sich gegenüber dem Mobilteil als valide Basisstation und gegenüber der Basisstation als valides Mobilteil ausgeben. Beiden Geräten kann er Befehle im Namen der jeweils anderen Partei senden. Zudem ist ein Angreifer in der Lage, wenn er die Kommunikation des *Authentifizierung eines PP* Szenarios abhört, den für die nachfolgende Kommunikation verwendeten kurzlebigen *Derived Cipher Key* zu berechnen. Mit diesem Schlüssel kann der Angreifer die nachfolgende Kommunikation entschlüsseln, manipulieren und weitere Nachrichten injizieren.

Da ein Angreifer schon während des *Easy Pairing* die korrekten Responses berechnen kann, kann er sich sogar in eine persistente Man-in-the-Middle-Position begeben. Ein Angreifer kann sich gegenüber der Basisstation als Mobilteil und gegenüber dem Mobilteil als Basisstation ausgeben und für beide Verbindungen ein separates *Easy Pairing* durchführen. Auf diese Weise speichern die Kommunikationsteilnehmer kein korrektes gemeinsames Geheimnis. Der gespeicherte UAK ist jeweils nur der Basisstation und dem Angreifer bzw. dem Mobilteil und dem Angreifer bekannt.

Im Falle eines Man-in-the-Middle Angriffs kann ein Angreifer zu einem späteren Zeitpunkt sogar ohne aktiven Angriff die Kommunikation zwischen Basisstation und Mobilteil unterbinden. Die Geräte können sich nicht gegenseitig authentifizieren, da sie kein gemeinsames Geheimnis besitzen. Um eine erfolgreiche Kommunikation durchzuführen, müsste der Angreifer als eine Art Proxy agieren und die Nachrichten korrekt und unverfälscht weiterleiten. Er ist hierbei allerdings in der Lage alle Nachrichten zu manipulieren, zu verwerfen oder neue Nachrichten zu injizieren.

4.1.2 Downgrade Angriffe

Während des Pairings tauschen die beteiligten Geräte im Protokollszenario *Obtain Access Rights* ihre bevorzugten Verschlüsselungs- und Authentifizierungsalgorithmen aus. Die Verbindung, über die diese Daten ausgetauscht werden, ist jedoch noch unauthentifiziert und unverschlüsselt. Hieraus entsteht die Gefahr, dass ein Angreifer die angegebenen Algorithmen manipuliert, um einen Downgrade Angriff durchzuführen.

Bei jeder neuen Verbindung zwischen einer Basisstation und einem Mobilteil wird der dabei zu verwendende Authentifizierungsalgorithmus erneut festgelegt. Wenn ein Angreifer während des *Obtain Access Rights* Szenarios die bevorzugten Algorithmen also noch nicht manipuliert hat, kann er auch zu Beginn der Authentifizierung einer neuen Verbindung einen Downgrade Angriff durchführen. Er kann unauthentifiziert eine DSAA2-Authentifizierungsanfrage im Namen des Kommunikationspartners ablehnen und als Grund angeben, dass der angegebene Authentifizierungsalgorithmus nicht unterstützt wird.

Um der ULE Spezifikation zu entsprechen, müssen alle Geräte neben dem – nach aktuellem Stand sicheren – DSAA2 Algorithmus auch den veralteten DSAA Algorithmus implementieren. Ein Downgrading auf den DSAA Algorithmus ist demnach auf jeden Fall möglich. Im Falle der Zurückweisung einer Authentifizierungsanfrage ist es deshalb sehr wahrscheinlich, dass die Authentifizierung erneut mit dem unsichereren DSAA Algorithmus initiiert werden würde.

Wenn ein Angreifer den verwendeten Authentifizierungsalgorithmus zu DSAA ändert, verwenden beide Partner nur Schlüssel mit einer Länge von 64 bit. Zudem können die von der deDECTed-Projektgruppe veröffentlichten Angriffe gegen den DSAA (siehe [DedectedDSAA]) durchgeführt werden. Wenn ein Angreifer den Algorithmus bricht, kann er die nachfolgende Kommunikation mitlesen, manipulieren und Nachrichten injizieren.

4.1.3 Fehlende Authentifizierung bei Verschlüsselung auf dem MAC-Layer mit DSC2

Ein bemerkenswerter Aspekt der Verschlüsselungsalgorithmen DSC und DSC2 ist, dass diese ausschließlich zu Datenschutz-Zwecken definiert worden sind. Eine Verschlüsselung der Kommunikation mit dem DSC2 Algorithmus schützt also nicht vor einer Manipulation der Nachrichten durch einen Angreifer.

Erst seit 2013 ist zusammen mit der ersten ULE Spezifikation in der Version v2.5.1 der DECT Data Link Layer Spezifikation der sogenannte LU14-Dienst definiert worden, der mit AES-CCM die Verwendung einer authentifizierten Verschlüsselung ermöglicht.[1] Wenn sich Geräte nicht vollständig an die ULE Spezifikation halten und ggf. nur die *alte* Verschlüsselung auf dem MAC-Layer mit DSC2 oder sogar DSC verwenden, ist theoretisch die gesamte Kommunikation durch einen Angreifer manipulierbar.

4.1.4 Komplexität der Dokumentenstruktur

Neben den obigen sehr konkreten Problemen besteht noch eine etwas abstraktere Problematik, die aus dem Umfang und der Komplexität der für ULE relevanten Spezifikationsdokumente hervorgeht. Eine vollständige Implementierung von ULE bedarf der Kenntnis einer sehr umfangreichen und heterogenen Menge an verschiedenen Dokumenten. Im Kontext von ULE handelt es sich dabei konkret um

- sieben Dokumente der DECT Spezifikation EN 300 175 (1229 Seiten), zum Teil jedoch nur Teile daraus,
- zwei Dokumente der ULE Spezifikation TS 102 939 (277 Seiten) und
- Teile von einigen weiteren Spezifikation zu DECT Profilen wie GAP (154 Seiten) und DPRS (287 Seiten).

Jedes dieser Dokumente existiert in einigen verschiedenen Versionen.

Wenn nun eine DECT-Basisstation entwickelt wird, die mehrere DECT-Profile unterstützen soll, steigert sich dieser Umfang weiter. Es müssen sehr viele unterschiedliche Funktionen implementiert werden und viele unterschiedliche Regeln einzelner Profile eingehalten werden. Das ULE Profil definiert beispielsweise einen eigenen NWK-Layer und auch im MAC-Layer sind ULE-spezifische Anpassungen vorgenommen worden.

[1] Die aktuelle Version der Data Link Layer Spezifikation ist v2.7.1 aus 2017.

Im Rahmen dieser Arbeit wurde bereits eine Inkonsistenz verschiedener Dokumente innerhalb der Beschreibung eines Protokollszenarios entdeckt: Im Dokument *Part 5: Network (NWK) layer* wird definiert, dass während der *Key Allocation* mit dem DSAA2-Algorithmus, zwischenzeitlich eine neue Challenge von der Basisstation generiert werden soll. Die Definition desselben Algorithmus im Dokument *Part 7: Security features* fordert dies nicht. Es ist nicht auszuschließen, dass gerade im Zusammenspiel unterschiedlicher DECT Profile noch weitere Inkonsistenzen bzw. Inkompatibilitäten auftreten. Einige dieser Probleme können auch sicherheitsrelevant sein.

4.2 Beispielhafte Angriffsszenarien

Im Kontext eines realen ULE Netzes folgen aus den Schwachstellen zwei unmittelbare Gefahren, die im Folgenden anhand von Beispielen verdeutlicht werden sollen.

4.2.1 Passiver Angriff auf das Pairing

Während in einem ULE Netzwerk beispielsweise ein smarter Rauchmelder eingerichtet wird, kann ein Angreifer das *Pairing* dieses Gerätes mitlesen. Nachdem der Rauchmelder installiert ist, kann der Angreifer aus den mitgelesenen Daten den Authentication Key der Verbindung zwischen dem Rauchmelder und der Basisstation berechnen. Auch, wenn nicht das Easy Pairing mit dem statischen Schlüssel 0000 verwendet wird, kann ein Angreifer offline den korrekten Authentication Key im Mittel in 2^{31} Versuchen erraten. Hierbei handelt es sich um einen praktisch umsetzbaren Angriff auf das Verfahren. Mit Kenntnis des Authentication Key kann der Angreifer im Namen des Rauchmelders einen Feueralarm auslösen. In der ULE Spezifikation wird sogar davon ausgegangen, dass der Alarm automatisiert von der DECT Basisstation an den Notruf weitergegeben werden kann, was wiederum einen unnötigen personalintensiven Einsatz zur Konsequenz haben kann.

Im *Global Home control and domotic* Umfeld ist ähnlich wichtig, dass man den Nachrichten, die im Namen eines Geräts versendet werden, vertrauen kann. Wenn zum Beispiel mit ULE steuerbare Rollladen installiert werden und diese abends heruntergefahren werden sollen, ist es fatal, wenn das System meldet, dass die Rollladen heruntergefahren wurden, ohne dass dies tatsächlich erfolgt ist.

4.2.2 Downgrade Angriff während der Authentifizierung

Wenn ein Angreifer das Pairing eines Gerätes nicht mitgelesen hat, kann er im nachträglich bei einem Verbindungsaufbau den zu verwendenden Authentifizierungsalgorithmus zum schwächeren DSAA Algorithmus ändern.

Wenn im Kontext des *Energy and appliances management* ein Stromzähler Daten an die Basisstation übertragen möchte, kann ein Angreifer hier die Authentifizierung schwächen, indem es beide Partner zwingt, den DSAA-Algorithmus zu verwenden.

Wenn der DSAA-Algorithmus verwendet wird, kann der Angreifer die mitgelesene Authentifizierung ggf. im Nachhinein brechen und den daraus abgeleiteten Schlüssel berechnen. Mit diesem Schlüssel kann ein Angreifer die aufgezeichneten verschlüsselten Daten nachträglich entschlüsseln.

Daten von vernetzten Stromzählern sind für Angreifer besonders interessant, da sie einen sehr guten Überblick über die Aktivitäten im Haus geben. Insbesondere wurde im Jahr 2012 zum Beispiel gezeigt, dass aus Stromzähler-Daten sogar Rückschlüsse auf das Fernsehprogramm gezogen werden können. [DaPriM2012]

4.3 Ausnutzbarkeit

Um das ULE Protokoll anzugreifen und auch die hier beschriebenen theoretischen Schwachstellen auszunutzen, kann ein Software Defined Radio (SDR) verwendet werden. Die PHL- und MAC-Layer von DECT können mithilfe von GNURadio implementiert werden. Um die zehn DECT-Channels gleichzeitig zu verarbeiten ist lediglich ein SDR notwendig, das eine Samplerate 20 MHz unterstützt, eine Bandbreite von 20 MHz besitzt und das Frequenzband um 1880 MHz − 1900 MHz empfangen kann. Für Offline-Angriffe gegen das Protokoll wird zudem eine Festplatte benötigt, die die empfangenen Samples ausreichend schnell speichern kann. Mit einer vervollständigten und um Angriffsmethoden ergänzte Software ist es damit sogar einem Laien ohne entsprechenden Kenntnisse in den Bereichen IT-Security und Signalverarbeitung und insbesondere ohne professionelle Vorbereitung (wie die Entwicklung oder Beschaffung DECT/ULE-spezifischer Hardware) möglich, Angriffe gegen ULE durchzuführen. Hierzu ist lediglich quelloffene Software notwendig, sodass neben generischer Hardware keine weiteren Kosten auf einen Angreifer zukommen.

Die Softwarelösung, mit der Angriffe gegen ULE durchgeführt werden können, hat im Bereich der Signalverarbeitung noch erhebliches Optimierungspotential. Die Funktionalität der Signalverarbeitung ist zwar bereits vollständig, wobei die einzelnen Teilschritte dieser Verarbeitung in ihrer Performance erheblich verbessert werden könnten. Für die Parsing- und Analysesoftware ist eine solide Basis geschaffen worden, die bereits den MAC-Layer und Teile des DLC- und NWK-Layers von ULE implementiert.

Aufbauend auf der entwickelten Software lassen sich bereits Daten von ULE-Netzen mitlesen und beispielsweise Verhaltensmuster von Nutzern erstellen. Hierzu müsste lediglich analysiert werden, wann welche Geräte kommunizieren.

A Glossar

Authentication Key Gemeinsames Geheimnis, das für die Authentifizierung beider Partner verwendet wird. Deterministisch aus dem UAK oder dem AC berechnet. In der Regel als K bezeichnet.

DECT Standard Authentication Algorithm Alter A Algorithm, der nicht öffentlich spezifiziert wurde

DECT Standard Authentication Algorithm #2 Neuer A Algorithm auf Basis von AES-128

DECT Standard Cipher Alter MAC-Verschlüsselungsalgorithmus, der nicht öffentlich spezifiziert wurde

DECT Standard Cipher #2 Neuer MAC-Verschlüsselungsalgorithmus auf Basis von AES-128

Fixed Part Eine DECT Basisstation

Key Allocation Protokollszenario, bei dem der UAK beider Partner aktualisiert wird, *siehe **UAK***

Nested Procedure Eine Prozedur, die während einer anderen Prozedur („verschachtelt“) ausgeführt wird, wird *Nested Procedure* genannt. Die andere Prozedur pausiert in dieser Zeit und wartet bspw. auf die Ergebnisse der *Nested Procedure*

Portable Part Ein ULE Sensor oder Aktor bzw. ein DECT *Mobilteil*

Protocol Data Unit Ein vom DLC-Layer erzeugtes Fragment eines Applikationsdatenpaketes

Protokollszenario Sicherheitsrelevanten Aktion, die durch das Protokoll durchgeführt werden kann

Service Data Unit Ein Applikationsdatenpaket, wie es der DLC-Layer entgegennimmt

Wireless Relay Station Ein Repeater für DECT

B Abkürzungen

AC Authentication Code

DCK Derived Cipher Key

DECT Digital Enhanced Cordless Telecommuniations

DLC Data Link Control

DSAA2 DECT Standard Authentication Algorithm #2

DSAA DECT Standard Authentication Algorithm

DSC2 DECT Standard Cipher #2

DSC DECT Standard Cipher

FP Fixed Part

GFSK Gaussian Frequency Shift Keying

IPUI International Portable User Identity

KS 128 bit intermediate Key (bei DSAA nur 64 bit), der im *A11*-Prozess aus dem Authentication Key und *RS* berechnet wird

KS′ 128 bit intermediate Key (bei DSAA nur 64 bit), der im *A21*-Prozess aus dem Authentication Key und *RS* berechnet wird

MAC Media Access Control

NWK Network

PARK Portable Access Rights Key

PDU Protocol Data Unit

PHL Physical Layer

PP Portable Part

PVC Permanent Virtual Circuit

RAND_F Vom FP generierte Challenge

RAND_P Vom PP generierte Challenge

RES1 Response des PP auf die Challenge des FP

RES2 Response des FP auf die Challenge des PP

RFPI Radio Fixed Part Identifier

RS 128 bit Zufallswert, der während der Authentifizierung genutzt wird um aus dem Authentication Key den *KS* abzuleiten

SDR Software Defined Radio

SDU Service Data Unit

TDMA Time Division Multiple Access

UAK User Authentication Key

ULE Ultra Low Energy

WRS Wireless Relay Station

C Literatur

[DaPriM2012] daprim. *Identifikation von Videoinhalten über granulare Stromverbrauchsdaten*. abgerufen am 01.01.2019. 2016. URL: http://www.daprim.de/?p=170.

[Dedected] Andreas Schuler u. a. *deDECTed.org*. abgerufen am 15.10.2018. URL: https://dedected.org.

[DedectedDSAA] Stefan Lucks u. a. *Attacks on the DECT authentication mechanisms*. abgerufen am 15.10.2018. 2009. URL: https://dedected.org/trac/attachment/wiki/DSAA-Analysis/Attacks%20on%20the%20DECT%20authentication%20mechanisms.pdf.

[DedectedDSC] Karsten Nohl, Erik Tews und Ralf-Philipp Weinmann. *Cryptanalysis of the DECT Standard Cipher*. abgerufen am 15.10.2018. 2010. URL: https://dedected.org/trac/attachment/wiki/DSC-Analysis/FSE2010-166.pdf.

[EN 300 175-1] Technical Specification ETSI EN 300 175-1 Version v2.7.1. *Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 1: Overview*. 2017. abgerufen am 02.03.2018. URL: https://www.etsi.org/deliver/etsi_en/300100_300199/30017501/02.07.01_60/en_30017501v020701p.pdf.

[EN 300 175-2] Technical Specification ETSI EN 300 175-2 Version v2.7.1. *Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 2: Physical Layer (PHL)*. 2017. abgerufen am 02.03.2018. URL: https://www.etsi.org/deliver/etsi_en/300100_300199/30017502/02.07.01_60/en_30017502v020701p.pdf.

[EN 300 175-3] Technical Specification ETSI EN 300 175-3 Version v2.7.1. *Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 3: Medium Access Control (MAC) layer*. 2017. abgerufen am 02.03.2018. URL: https://www.etsi.org/deliver/etsi_en/300100_300199/30017503/02.07.01_60/en_30017503v020701p.pdf.

[EN 300 175-4] Technical Specification ETSI EN 300 175-4 Version v2.7.1. *Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 4: Data Link Control (DLC) layer*. 2017. abgerufen am 02.03.2018. URL: https://www.etsi.org/deliver/etsi_en/300100_300199/30017504/02.07.01_60/en_30017504v020701p.pdf.

[EN 300 175-5] Technical Specification ETSI EN 300 175-5 Version v2.7.1. *Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 5: Network (NWK) layer*. 2017. abgerufen am 02.03.2018. URL: https://www.etsi.org/deliver/etsi_en/300100_300199/30017505/02.07.01_60/en_30017505v020701p.pdf.

[EN 300 175-6] Technical Specification ETSI EN 300 175-6 Version v2.7.1. *Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 6: Identities and addressing*. 2017. abgerufen am 02.03.2018. URL: https://www.etsi.org/deliver/etsi_en/300100_300199/30017506/02.07.01_60/en_30017506v020701p.pdf.

[EN 300 175-7] Technical Specification ETSI EN 300 175-7 Version v2.7.1. *Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 7: Security features*. 2017. abgerufen am 02.03.2018. URL: https://www.etsi.org/deliver/etsi_en/300100_300199/30017507/02.07.01_60/en_30017507v020701p.pdf.

[EN 300 444] Technical Specification ETSI EN 300 444 Version v2.5.1. *Digital Enhanced Cordless Telecommunications (DECT); Generic Access Profile (GAP)*. 2017. abgerufen am 23.10.2018. URL: https://www.etsi.org/deliver/etsi_en/300400_300499/300444/02.05.01_60/en_300444v020501p.pdf.

[EN 300 700] Technical Specification ETSI EN 300 700 Version v2.1.1. *Digital Enhanced Cordless Telecommunications (DECT); Wireless Relay Station (WRS)*. 2017. abgerufen am 02.03.2018. URL: https://www.etsi.org/deliver/etsi_en/300700_300799/300700/02.01.01_60/en_300700v020101p.pdf.

[gr-dect2] ReDECTed. *gr-dect2*. abgerufen am 15.10.2018. 2015. URL: https://github.com/pavelyazev/gr-dect2.

[Lücke2010] Hayo Lücke. *AVM - Neues von der Fritz!Box*. abgerufen am 06.11.2018. 2010. URL: https://www.onlinekosten.de/news/avm-neues-von-der-fritzbox_177339.html.

[PanasonicKX-HN6011] Panasonic. *KX-HN6011 Home Safety Starter Kit*. abgerufen am 06.11.2018. 2018. URL: https://www.panasonic.com/de/consumer/smart-home/smart-home-produkte/KX-HN6011.specs.html.

[ReDECTed] Marc Newlin. *ReDECTed*. abgerufen am 15.10.2018. 2015. URL: http://haxpo.nl/haxpo2015ams/sessions/redected/.

[RFC 3610] Informational IETF RFC 3610. *Counter with CBC-MAC (CCM)*. 2003. abgerufen am 28.11.2018. URL: https://tools.ietf.org/html/rfc3610.

[STRIDE] Adam Shostack. *Threat Modelling – Designing for Security*. Wiley, 2014.

[Telkogeräte2016] Bundestag. *Gesetz zur Auswahl und zum Anschluss von Telekommunikationsendgeräten*. abgerufen am 06.11.2018. 2016. URL: https://www.bgbl.de/xaver/bgbl/start.xav?startbk=Bundesanzeiger_BGBl%5C&jumpTo=bgbl116s0106.pdf.

[Tews2012] Erik Tews. *DECT Security Analysis*. Technische Universität Darmstadt. abgerufen am 02.03.2018. 2012. URL: https://tuprints.ulb.tu-darmstadt.de/2932/.

[TS 102 939-1] Technical Specification ETSI TS 102 939-1 Version v1.3.1. *Digital Enhanced Cordless Telecommunications (DECT); Ultra Low Energy (ULE); Machine to Machine Communications; Part 1: Home Automation Network (phase 1)*. 2017. abgerufen am 02.03.2018. URL: https://www.etsi.org/deliver/etsi_ts/102900_102999/10293901/01.03.01_60/ts_10293901v010301p.pdf.

[TS 102 939-2] TECHNICAL SPECIFICATION ETSI TS 102 939-2 Version v1.2.1. *Digital Enhanced Cordless Telecommunications (DECT); Ultra Low Energy (ULE); Machine to Machine Communications; Part 2: Home Automation Network (phase 2)*. 2017. abgerufen am 02.03.2018. URL: https://www.etsi.org/deliver/etsi_ts/102900_102999/10293902/01.02.01_60/ts_10293902v010201p.pdf.

[UleSpecs] ULE Alliance. *Specifications*. ULE Alliance. abgerufen am 02.03.2018. URL: https://www.ulealliance.org/specifications.