HOCHSCHULE
FÜR ANGEWANDTE
WISSENSCHAFTEN
MÜNCHEN

Hochschule für angewandte Wissenschaften München

Fakultät für Informatik und Mathematik

Masterthesis in Informatik

Ein vom Verschlüsselungsverfahren unabhängiger Man-in-the-Middle Angriff in GSM

Sebastian Stumpf

University of Applied Sciences Munich

Department of computer science and mathematics

Master thesis in computer science

An encryption independent Man-In-The-Middle Attack in GSM

Sebastian Stumpf

Author: Sebastian Stumpf

Matrikelnummer: 49780514

Prüfer: Prof. Dr. Alf Zugenmaier

Semester: 5. Mastersemester Informatik

Abgabedatum: 06.06.2017

Zusammenfassung

In dieser Masterarbeit wird ein neuartiger Man-in-the-Middle (MitM) Angriff auf eine Sprachverbindung im Global System for Mobile Communications (GSM) Netz entwickelt. Der Angriff nutzt den fehlenden Integritätsschutz auf der Funkschnittstelle aus, um in den verschlüsselten Anrufaufbau eines Opfers einzugreifen und den Anruf an ein Mobiltelefon unter Kontrolle des Angreifers umzuleiten. Im theoretischen Teil der Arbeit wird der Angriff entwickelt, seine Machbarkeit mathematisch nachgewiesen und die zugrunde liegende Schwachstelle beschrieben. Im praktischen Teil wird der Angriff für einen MitM, in einer Testumgebung mit einer virtualisierten Funkschnittstelle, implementiert und durchgeführt. Die in GSM verwendeten Verschlüsselungsverfahren sind Stromchiffren, die die Vertraulichkeit des Datenstroms durch die XOR-Kombination mit einem Schlüsselstrom schützen. Es wird gezeigt dass bekannte Teile des Chiffrestroms beliebig manipuliert werden können. Da GSM kein Verfahren für den Integritätsschutz spezifiziert, kann der Angreifer die Telefonnummer im ausgehenden Anruf also unbemerkt ersetzen und den Anruf an ein von ihm bestimmtes Endgerät umleiten. Durch die Verknüpfung des eingehenden Anrufs mit einem neuen Anruf bei der ersetzten Nummer, erhält der Angreifer Zugriff auf das geführte Gespräch – ein MitM-Angriff auf die Sprachverbindung. Das Netzwerk und die Endgeräte kümmern sich für den Angreifer um Verschlüsselung und Kodierung der Sprachdaten, da es sich um reguläre GSM-Anrufe handelt. Der Angriff ist selbst bei der Verwendung von sicheren Verschlüsselungsverfahren, wie A5/4 anwendbar, da die Verschlüsselung nicht gebrochen werden muss. Auch das aktuelle 3G Authentication and Key Agreement (AKA) bietet keinen Schutz, da es in GSM zwar die gegenseitige Authentifizierung ermöglicht, aber keinen Integritätsschutz unterstützt. Der Angriff erfordert die teilweise Kenntnis der vom Opfer angerufenen Telefonnummer („Known Plaintext“). Diese kann als gegeben angenommen werden, da die Rufnummern in der Regel bekannte Teile aufweisen. Meyer und Wetzel [2004] zeigten, dass es möglich ist, ein MitM-Gerät auf der Funkschnittstelle zu installieren, was für die Arbeit vorausgesetzt und nicht näher untersucht wird. Für die praktische Durchführung des Angriffs wird die physikalische Ebene der Funkschnittstelle, auf Basis von Multicast-Sockets, virtualisiert und für diese ein MitM implementiert. Die Durchführung des Angriffs erfolgt im Rahmen des Osmocom Projekts. Im letzten Teil der Arbeit wird Bezug zu verwandten Angriffen auf den GSM-Standard aufgebaut und die Vorteile gegenüber diese herausgearbeitet.

Abstract

In this master thesis, a new Man-in-the-Middle (MitM) attack on a voice connection in the Global System for Mobile Communications (GSM) network is being developed. The attack exploits the lack of integrity protection on the radio interface to manipulate a victim's outgoing call and redirect it to a mobile phone designated by the attacker. In the theoretical part of the thesis, the attack and the security flaws it is based on, as well as relations to current attacks are established. In the practical part, the call setup manipulation is verified by a MitM on a virtual radio interface. The encryption methods used in GSM are stream ciphers. This means the data stream is secured by combining it with a key stream using the exclusive-or (XOR) operation. It is shown that a cipher stream generated by a stream cipher can be arbitrarily manipulated if the plaintext is known. Since GSM does not specify a procedure for integrity protection, the called phone number in the outgoing call can be replaced by the attacker. The attacker can redirect the call to a mobile entity under control and thus receives the voice traffic from the calling victim. By linking this voice traffic with a new call to the original called number, the attacker creates a MitM in between the two communication partners. Because incoming and outgoing calls are valid GSM calls, the network and mobile devices take care of the encryption and encoding of the voice data by themselves. This means the attacker has access to the unencrypted communication and can record or manipulate the call as they wish. Since the encryption does not have to be broken, the attack is working even with strong encryption algorithms. Furthermore, the use of the more recent 3G Authentication and Key Agreement (AKA) instead of the outdated 2G AKA provides no protection from the attack. It brings mutual authentication to GSM networks, but does not ensure integrity between Mobile Station (MS) and Base Transceiver Station (BTS). The attack requires partial known plaintext of the called mobile number, which can be assumed in GSM. Mobile phone numbers usually have known parts, like the Network Destination Code (NDC) or the Call Control (CC). The installation of a MitM on the physical level of the real radio frequency interface is not investigated in this work. However, the physical level of the radio interface is virtualized and the MitM is inserted into it for the verification of the attack in the practical part. The implementations of the virtual physical interface as well as the MitM are carried out within the framework of the Osmocom project.

An dieser Stelle möchte ich all jenen danken, die durch ihre Unterstützung zu dieser Masterarbeit beigetragen haben.

Harald Welte, für die Einladung zu OsmoCon2017 und OsmoDevCon2017 und die Beantwortung vieler Fragen rund um die Implementierung der virtuellen Um-Schnittstelle.

Prof. Alf Zugenmaier, für das Teilen der Idee hinter dem Angriff und die regelmäßigen Treffen, in denen er mir mit Rat beiseite stand.

Meiner Freundin Carmen, für ihr immer offenes Ohr und ihre vielen nützlichen Tipps.

Und nicht zuletzt den Lesern der Arbeit für ihre Zeit und ihr Feedback.

Inhaltsverzeichnis

1. Einleitung

Seit der Einführung von Global System for Mobile Communications (GSM) 1992 hat sich der Mobilfunk stetig weiterentwickelt. So wurden neue, schnellere und sicherere Standards für Mobilfunknetzwerke spezifiziert und ausgerollt. Weil das GSM-Netz aber wegen seiner bestehenden und weit verbreiteten Infrastruktur flächendeckenden Empfang liefert und kostengünstig ist, wird es in den meisten Regionen trotz Ausbau der 3G und 4G Netze weiterhin unterstützt und verwendet [opensignal.com, 2017]. In Japan, Korea und Singapur ist die Abschaltung des 2G Netzes bereits erfolgt und in Australien bis September 2017 geplant. Das Ziel dabei ist vor allem die Wiederverwendung der begehrten, aber von GSM belegten Frequenzen im 900 MHz Band. In Europa wird laut heise.de [2016] aktuell die Unterstützung und Instandhaltung der Infrastruktur von GSM bis etwa 2020 vorausgesagt - viele Netzanbieter halten sich diesbezüglich bedeckt. Da vor allem im Internet of Things (IoT) Bereich und für die Machine to Machine (M2M) Kommunikation viele GSM-Module verwendet werden, die die neueren Standards nicht unterstützen, hält man sich in einigen Ländern wie Deutschland mit konkreten Angaben ganz zurück. Der Lebenszyklus dieser meist im Embedded-Bereich verwendeten Geräte ist in der Regel hoch und ohne GSM müssten sie ersetzt werden, was hohe Kosten verursachen würde. Auch 2017 wird noch in den Ausbau von GSM-Netzwerken investiert. In Mexiko wird zum Beispiel für schwer erreichbare Regionen von Non-Profit-Organisationen der GSM Netzausbau auf Basis von Open-Source-Software des Osmocom Projektes vorangetrieben [osmocom.org, 2017b]. Es wird derzeit angenommen, dass das 3G Netz als Übergangstechnologie vom sprachbasierten GSM zum datenbasierten Long Term Evolution (LTE) früher abgeschaltet wird als GSM. Die GSM-Infrastruktur wird, wenn man von der derzeitigen Entwicklung ausgeht, also noch lange erhalten bleiben. Die folgenden Zitate stützen diese Behauptung.

„In terms of global reach, cellular networks already cover 90 percent of the world's population. WCDMA and LTE are catching up, but GSM will offer superior coverage in many markets for years to come.“ [Ericsson, 2016]

1. Einleitung

„Speziell für M2M Anwendungen könnte GSM über 2020 hinaus weiterhin relevant bleiben." [heise.de, 2016, Tom Tesch]

„We are still maintaining our old networks, and modernising the network, also still delivering 2G and 3G services to the customer. And yes, 2G will continue longer than we expected" [mobileworldlive.com, 2015, Matthias Sauder, Vodafone]

Der wohl noch länger andauernden Laufzeit von GSM stehen dessen veraltete Sicherheitsmechanismen und die Einstellung von Mobilfunkanbietern und Herstellen, diese zu überarbeiten gegenüber. Harald Welte fasst diese in seinem Blog zum Thema Sicherheit zusammen.

„GSM equipment manufacturers and mobile operators have shown no interest in fixing gaping holes in their security system." [laforge.gnumonks.org, 2010, Harald Welte]

So wird GSM ohne größere Veränderungen auf dem gleichen Stand der Technik wie vor 30 Jahren betrieben. Trotz einiger Versuche die Sicherheit zu verbessern, wie die Einführung der gegenseitigen Authentifizierung oder die Spezifikation von neuen Verschlüsselungsalgorithmen wie A5/4, bleibt GSM verwundbar und anfällig für eine Vielzahl von Angriffen. Gründe dafür sind Schwachstellen in den Anpassungen der Sicherheitsmechanismen und die verzögerte Einführung dieser von den Netzanbietern. Das auch für GSM spezifizierte Universal Mobile Telecommunications System (UMTS)-Authentication and Key Agreement (AKA) gewährleistet in GSM zum Beispiel gegenseitige Authentifizierung, aber keinen Integritätsschutz, eine Schwachstelle in der Spezifikation. Der in dieser Arbeit vorgestellte Man-in-the-Middle (MitM)-Angriff nutzt diese Schwachstelle, sowie in GSM unverschlüsselt preisgegebene Informationen, um in den Anrufaufbau einzugreifen und ein Telefonat zu einem Angreifer umzuleiten. Der Eingriff liegt in der Manipulation der angerufenen Telefonnummer in der verschlüsselten, für den Anrufaufbau zuständigen Signalisierungsnachricht. Der Angriff verdeutlicht die Notwendigkeit, Möglichkeiten für den Schutz der Integrität für den GSM-Standard zu spezifizieren und in den GSM-Netzwerken einzuführen. Verschlüsselung ohne Integritätsschutz stellt eine gravierende Schwachstelle dar. Vor allem im Hinblick auf die voraussichtlich noch lange Laufzeit von GSM, muss diese dringend behoben werden.

2. Ziel

Ziel dieser Arbeit ist es, einen MitM-Angriff auf eine verschlüsselte Sprachverbindung im GSM-Netz umzusetzen. Der Angriff soll den fehlenden Integritätsschutz der Kommunikation zwischen Netzteilnehmer und Netzwerk ausnutzen, um ein ausgehendes Telefonat an eine vom Angreifer bestimmte Telefonnummer umzuleiten. Die Verschlüsselung soll nicht gebrochen werden, der kryptografische Schlüssel wird also als unbekannt vorausgesetzt.

Der Angriff soll sowohl theoretisch entwickelt, als auch praktisch umgesetzt werden. Im theoretischen Teil soll die Möglichkeit der Identifizierung der für den Anrufaufbau zuständigen Nachricht im Nachrichtenfluss gezeigt und die Machbarkeit der Manipulation dieser Nachricht, ohne Kenntnis des kryptografischen Schlüssels, mathematisch nachgewiesen werden. Im praktischen Teil der Arbeit soll der Angriff innerhalb einer Testumgebung, anhand eines vom Opfer ausgehenden Anrufs, praktisch durchgeführt werden. Die Implementierung des Angriffs und dafür nötiger Anwendungen soll das Open-Source-Projekt Osmocom nutzen. Für den Aufbau der Testumgebung sollen die Projekte osmoBTS (die Base Transceiver Station (BTS)) und osmocomBB (die Mobile Station (MS)) über eine virtuelle Funkschnittstelle verbunden werden. Die virtuelle Funkschnittstelle soll die Übertragung der Nachrichten über User Datagram Protocol (UDP)/Internet Protocol (IP), statt einem Funksignal ermöglichen. Für die virtuelle Funkschnittstelle soll eine MitM-Anwendung implementiert werden, die Zugriff auf die Kommunikation zwischen MS und BTS hat. Der ausgehende Anruf des Opfers soll vom MitM an eine vom Angreifer bestimmte Rufnummer umgeleitet werden. Die Weiterleitung der Audiodaten, vom Mobilfunktelefon des Angreifers an die ersetzte Rufnummer, ist nicht Teil dieser Arbeit, ebenso wie die Umsetzung des MitMs-Angriffs auf der realen Funkschnittstelle.

Das übergreifende Ziel ist es, den Bedarf an Integritätsschutz generell und speziell in GSM zu verdeutlichen. Es soll gezeigt werden, dass die fehlende Integrität die Verschlüsselung nutzlos macht, wenn ein Angreifer Teile der verschlüsselten Signalisierungsnachrichten kennt.

3. Grundlagen

Folgenden Abschnitte umfassen Grundlagen, die für das Verständnis der weiteren Kapitel der Arbeit von Nutzen sind. Es wird auf die GSM-Netzarchitektur, verschiedene von 3rd Generation Partnership Project (3GPP) spezifizierte Protokolle und Sicherheitsmechanismen in GSM eingegangen.

3.1. GSM Architektur

Die GSM-Netzarchitektur ist, wie in Abbildung 3.1 zu sehen, hierarchisch aufgebaut. Auf General Packet Radio Service (GPRS) Komponenten sowie Schnittstellen der Infrastruktur zu 3G und 4G Netzen wird nicht eingegangen. Ein Base Station Subsystem (BSS) ist zuständig für Verwaltung und den Betrieb der Funkschnittstelle die den verbundenen Endgeräten die Übertragung von Sprache und Daten ermöglicht. Die Funktion des Network Switching Subsystem (NSS) eines Netzbetreibers besteht in der Vermittlung von Gesprächs- oder Datenverbindungen innerhalb des eigenen BSS, oder zu Partnernetzwerken wie dem Festnetz und Mobilfunknetzen anderer Anbieter. Hier wegen mangelnder Relevanz nicht erklärte Begriffe und Komponenten finden sich im Anhang in Abschnitt B.1.

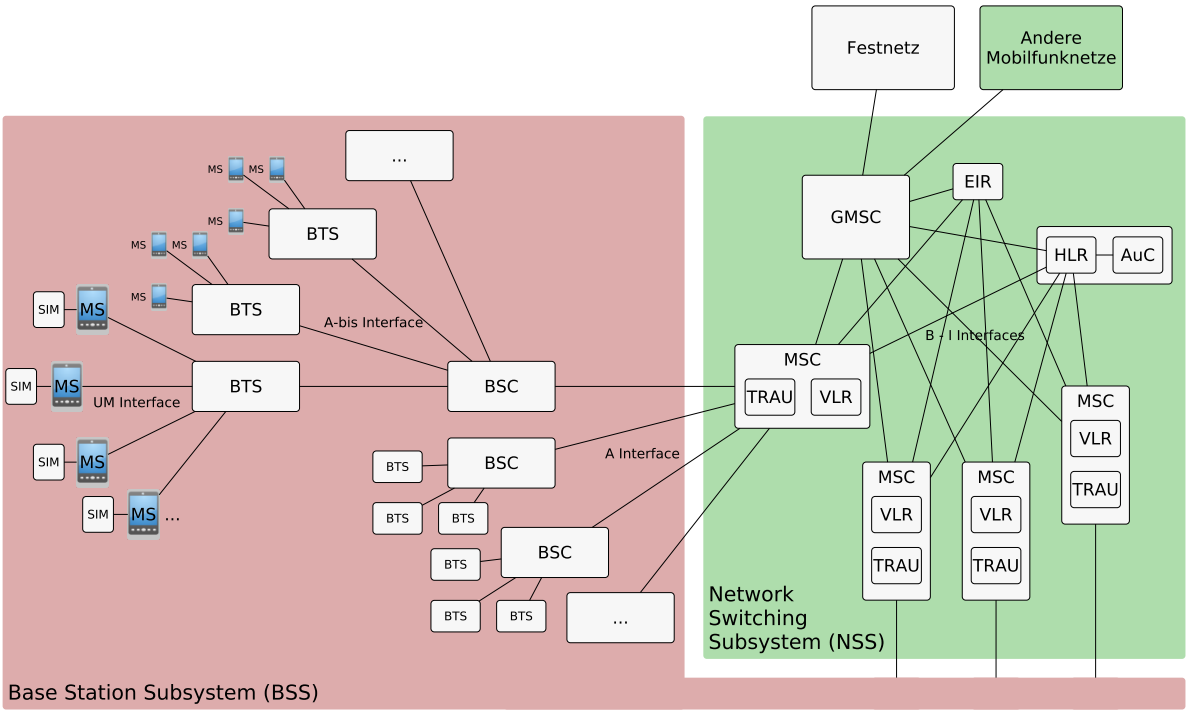


Abbildung 3.1.: Die GSM-Netzarchitektur, erstellt mit yEd nach [Schnabel, 2003]

MS - Mobile Station Die MS bezeichnet das mobile Endgerät, das Zugriff auf das Netzwerk erhält. Um die volle Funktionalität des Netzwerks nutzen zu können braucht man eine SIM-Karte. Auf ihr sind International Mobile Subscriber Identity (IMSI) und geheimer Schlüssel eines Netzteilnehmers gespeichert. Die vom Anbieter zugeteilte IMSI identifiziert den Nutzer eindeutig, womit ihm die verwendeten Dienste in Rechnung gestellt werden können. Das Subscriber Identity Module (SIM) Modul ist ein Mikrocontroller, der zusammen mit dem Authentication Center (AuC) Zugriff auf den geheimen Schlüssel Ki und die Algorithmen A3 (Authentifizierung) und A8 (Schlüsselgenerierung) hat. Neben IMSI und Ki können auch Benutzerdaten wie persönliche Kontakte und der Anrufverlauf gespeichert werden.

BTS - Base Transceiver Station Die BTS kommuniziert über die Funkschnittstelle auf einer oder mehreren zugewiesenen Trägerfrequenzen mit den MS's. Dabei werden Uplink- und Downlinkfrequenz von einer Absolute Radio Frequency Channel Number (ARFCN) definiert. Damit sich benachbarte BTS nicht gegenseitig stören, werden ihnen sich nicht überlappende Trägerfrequenzen zugeteilt. Mehrere MS's können gleichzeitig mit einer BTS verbunden sein.

BSC - Base Station Controller Der BSC kann über das Abis Interface auf dem Operation and Maintenance Link (OML) mehrere BTS's verwalten und vermittelt Daten vom NSS an den zuständigen BTS. Nutzdaten von von Mobiltelefonen werden an das NSS weitergeleitet.

MSC - Mobile Switching Center Mehrere BSCs sind über ein MSC mit dem Kernnetzwerk und anderen Netzwerken verbunden. Das MSC ist Teil des Signaling System 7 (SS7) Netzwerks, hat also Zugriff auf alle Komponenten darin. Das Visitor Location Register (VLR) ist meist direkt im MSC integriert.

HLR - Home Location Register Das HLR enthält eine Datenbank mit Informationen aller Netzteilnehmer des Providers sowie das AuC mit den Algorithmen A3 und A8. Gespeichert sind Vertragsdetails, Zugriffsberechtigungen auf Netzdienste, aktuelles Guthaben, Ki, Mobile Subscriber Routing Number (MSRN), aktuelles VLR und Mobile Subscriber ISDN Number (MSISDN) zu jeder IMSI. Die MSRN ermöglicht das Auffinden des Teilnehmers in fremden Netzen beim Roaming. Das aktuelle VLR wird für die Lokalisierung im eigenen Netz verwendet. Falls sich beim Location Area Update (LAU) eine MS aus dem Zuständigkeitsbereich des MSC entfernt, muss im HLR die Adresse des VLR angepasst werden.

VLR - Visitor Location Register Der zeitaufwendige Zugriff des MSC auf das HLR über das SS7 wird durch das VLR umgangen. Es enthält eine Kopie der Daten des HLR und zusätzlich die Felder Temporary IMSI (TMSI) und Location Area Index (LAI). Mit dem LAI kann der BSC, in dessen Zuständigkeitsbereich sich der Teilnehmer gerade befindet, identifiziert werden. Die Teilnehmerdaten sind über TMSI, IMSI und die MSRN indiziert. Die Adresse des HLR wird benötigt um Authentifizierungsanfragen an dieses weiterzuleiten.

Um-Schnittstelle (siehe Abschnitt 3.3) Die Funk- oder Radioschnittstelle wird für den Datentransfer zwischen MS und BTS verwendet. Frequency Division Multiple Access (FDMA) ermöglicht die Kommunikation mit verschiedenen BTS und legt die Richtung des Datenflusses (Uplink / Downlink) fest. Mit Time Division Multiple Access (TDMA) werden verschiedene Kommunikationskanäle mit einer BTS definiert, durch die verschiedene Typen von Signalisierungsinformationen und Daten unterschieden werden können. Übertragene Nachrichten sind kodiert und in der Regel verschlüsselt.

3.2. GSM Signalisierungsprotokolle

Logische Kanäle (siehe Tabelle 3.1) können entweder Signalisierungsinformationen oder Nutzdaten übertragen. Wegen der unterschiedlichen Anforderungen werden in GSM dafür zwei Protokollstapel definiert - Signaling Plane und User Plane [Eberspächer u. a., 2008, Kap. 5.2, 5.3].

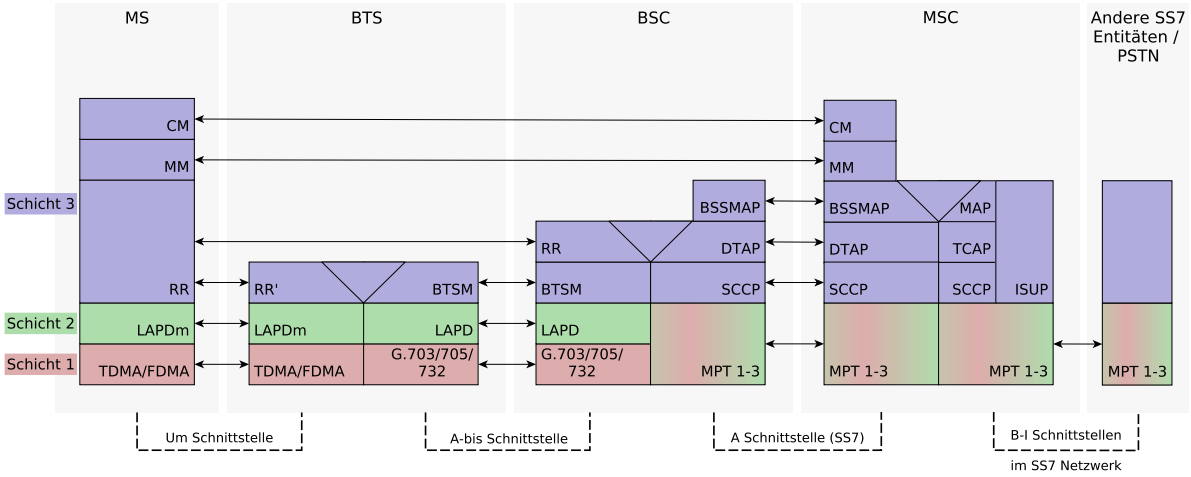


Abbildung 3.2.: Protokollstapel für Signalisierung in GSM, erstellt mit yEd nach [Eberspächer u. a., 2008]

Abbildung 3.2 bietet eine Übersicht über die Signaling Plane, deren Protokollstruktur und Schnittstellen in den folgenden Kapiteln erklärt werden. Hier wegen mangelnder Relevanz nicht erklärte Protokolle und Schnittstellen finden sich im Anhang in Abschnitt B.2.

3.3. Um Schnittstelle

Die Funkschnittstelle zwischen MS und BTS wird Um-Schnittstelle oder auch kurz Um genannt. Die Frequenzen für Uplink und Downlink, über die mit einer BTS kommuniziert werden kann, werden über die ARFCN der BTS bestimmt. Auf dem Uplink läuft der Datenfluss Richtung BTS, auf dem Downlink Richtung MS. Da die Übertragung über die Funkschnittstelle fehlerbehaftet ist, müssen für eine zuverlässige Verbindung Fehlerkorrekturmechanismen implementiert sein.

Das Übertragungsprotokoll der physikalischen Schicht basiert auf einer Kombination aus FDMA und TDMA (siehe Unterabschnitt B.2.1). Durch FDMA wird das verfügbare Frequenzband in verschiedene Trägerfrequenzen unterteilt. Mit TDMA werden auf einer Trägerfrequenz acht Zeitschlitze definiert und physikalischen Kanälen zugeteilt. Einem physikalischen Kanal wird durch Multiframes wiederum eine sich wiederholende Sequenz logischer Kanäle zugewiesen. Logische Kanäle bieten verschiedene Funktionen an. Auf Traffic Channels (TCHs) werden zum Beispiel Sprachdaten, auf Broadcast Control Channels (BCCHs) Broadcast Signalisierungsdaten und auf Common Control Channels (CCCHs) Signalisierungsdaten für einzelne Teilnehmer übertragen.

Im Zeitschlitz eines physikalischen Kanals kann genau die Datenmenge eines Bursts übertragen werden. In Abbildung 3.3 ist der Aufbau eines normalen Bursts dargestellt. Normale Burst werden in GSM sowohl für die Übertragung von Nutzdaten auf TCHs, als auch für die Übertragung von Signalisierungsnachrichten auf Standalone Dedicated Control Channel (SDCCH), Slow Associated Control Channel (SACCH), Fast Associated Control Channel (FACCH) und weiteren verwendet. Wie man in der Grafik sehen kann, trägt jeder Burst zwei 57 Bit Datenblöcke auf einmal. Traingssequenz, vorderer und hinterer Tail sind für die Funkübertragung relevant und für diese Arbeit nicht von Bedeutung. Auf die Stealing Flags wird im folgenden Abschnitt eingegangen. Der Aufbau verschiedener Bursttypen kann in [TS-05.02, Kap. 5.2.3] nachgelesen werden.

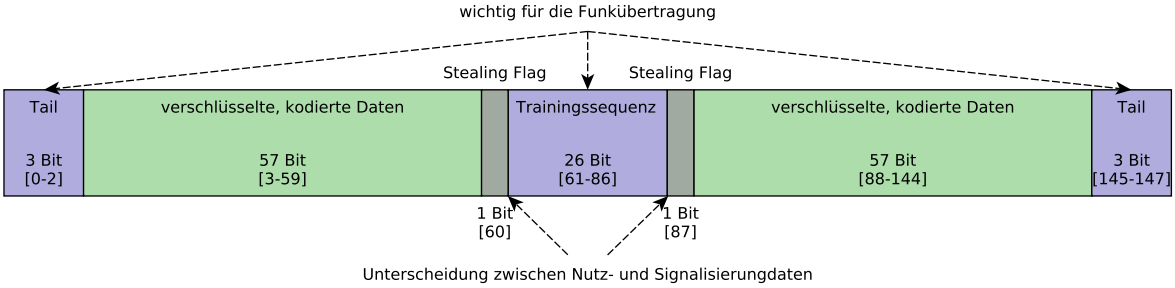


Abbildung 3.3.: Der Aufbau eines normalen Bursts, erstellt mit yEd nach [TS-05.02, Kap. 5.2.3]

3.3.1. Logische Kanäle – GSM-Multiframe

Physikalischen Kanälen können verschiedene Sequenzen logischer Kanäle zugeordnet werden[TS-04.03, Kap. 6].

Es wird ein sich wiederholendes Multiframe definiert, welches sich auf Signalisierungskanälen nach 51 Frames und auf Datenkanälen nach 26 Frames wiederholt. Die Struktur des Multiframes, also die Sequenz seiner logischen Kanäle, ordnet damit einer Nachricht, über ihre Frame Number (FN), genau einem logischen Kanal zu. Durch die unterschiedliche Größe der Multiframes für TCH, BCCH und CCCH beginnen diese erst nach ihrem gemeinsamen Vielfachen ($26 \cdot 51$ Frames) wieder zusammen von vorne – einem Superframe. Aufgrund der begrenzten Größe der FN wird ein Hyperframe von 2048 Superframes definiert, nach der die FN wieder bei 0 beginnt [TS-05.02, Kap. 6]. Die FN wiederholt sich also alle 3 Stunden, 28 Minuten und 53.76 Sekunden.

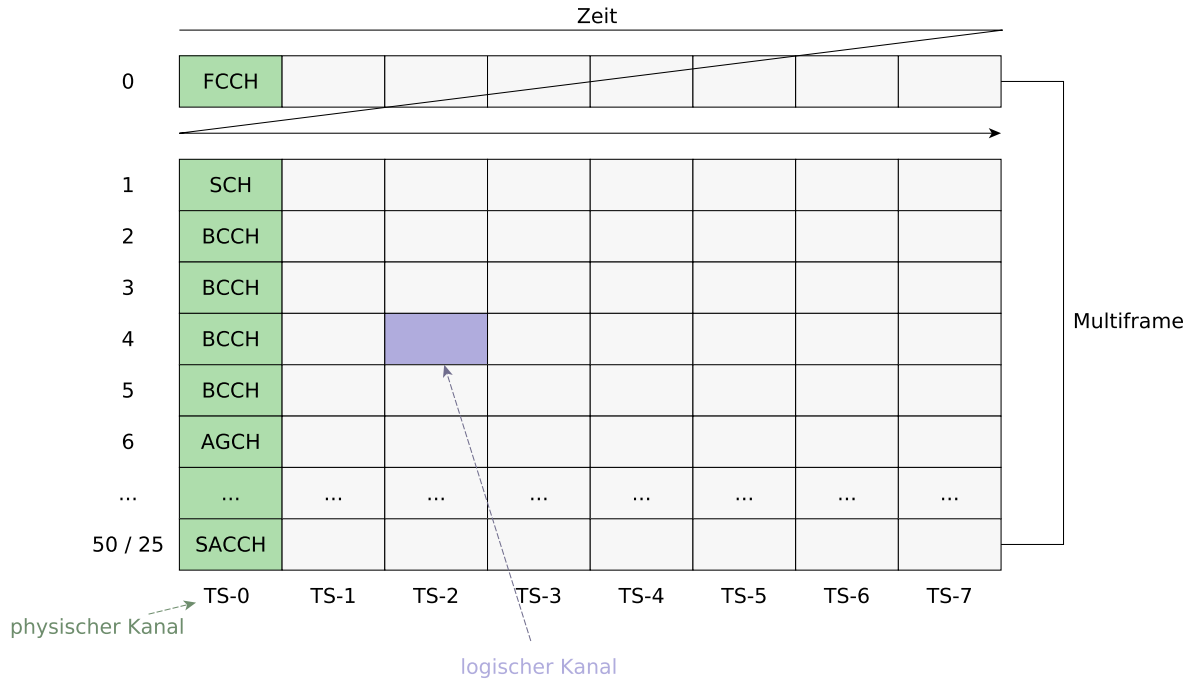


Abbildung 3.4.: Die GSM Multiframe Struktur, erstellt mit yEd nach [Sauter, 2011, Kap. 1.7.3]

In GSM wird jedem physikalischen Kanal ein Multiframe zugeordnet. Das Multiframe, das für die Bestimmung des logischen Kanals einer Nachricht angewendet wird, ist also

durch den Zeitschlitz bestimmt, auf dem die Nachricht übertragen wurde. Aus der FN, mit der die Nachricht empfangen wurde, wird das Offset im Multiframe berechnet und ihr damit ein logischer Kanal zugeordnet. Die FN und der physikalische Kanal bestimmen also eindeutig den logischen Kanal einer Nachricht. Tabelle 3.1 gibt eine Übersicht über die logischen Kanäle und ihre Funktionen.

Kanäle vom Typ BCCH, CCCH und Dedicated Control Channel (DCCH) übertragen Signalisierungsinformationen und sind Teil der Signaling Plane. Der TCH überträgt ausschließlich Sprachdaten und wird der User Plane zugeordnet. Der SDCCH wird für Signalisierung und im Fall von Short Message Service (SMS) Nachrichten auch für Nutzdaten verwendet, weshalb durch Multiplexing zwischen User und Signaling Plane unterschieden wird. Für das Multiplexing ist das SAP Identifier (SAPI) Feld des Link Access Procedure for the Dm-Channel (LAPDm) Headers zuständig (siehe Unterabschnitt 3.3.3).

Name	Beschreibung	Richtung	Typ
BCCH	Broadcast Control Channel	Downlink	Broadcast
SCH	Synchronization Channel	Downlink	Broadcast
FCCH	Frequency Correction Channel	Downlink	Broadcast
AGCH	Access Grant Channel	Downlink	Common Control
PCH	Paging Channel	Downlink	Common Control
RACH	Random Access Channel	Uplink	Common Control
FACCH	Fast Associated Control Channel	Bidirektional	Dedicated Control
SACCH	Slow Associated Control Channel	Bidirektional	Dedicated Control
SDCCH	Standalone Dedicated Control Channel	Bidirektional	Dedicated Control
TCH	Traffic Channel	Bidirektional	Nutzdatenkanal

Tabelle 3.1.: Die logischen Kanäle der Funkschnittstelle

Broadcast Control Channels (BCCHs) werden für Nachrichten von der BTS an alle sich im Empfangsbereich befindenden MS's verwendet. Es gibt sie deshalb nur im Downlink. [TS-04.03, Kap. 4.1.1]

- **BCCH**: Auf dem BCCH werden über die System Information (SI) Nachrichten regelmäßig Konfiguration und Parameter des Netzwerks verschickt. Davon gibt es insgesamt 9, von denen aber nur SI-1 bis SI-4 auf dem BCCH übertragen werden.

- **SCH**: Die auf diesem Kanal gesendeten Bursts ermöglichen es Endgeräten, sich mit der Multiframe-Struktur der BTS zu synchronisieren. Damit das MS den Burst dekodieren kann, obwohl es zu diesem Zeitpunkt die Entfernung und damit den genauen Anfang des empfangenen Bursts nicht kennt, haben die Synchronisation-Bursts ein spezielles Format.

- **FCCH**: Dieser Kanal dient zur Kalibrierung des Transceivers der MS und definiert den Anfang des Multiframes.

Auf den **Common Control Channels (CCCHs)** werden Signalisierungsinformationen ausgetauscht, mit denen dedizierte bidirektionale Kanäle beantragt und vergeben werden können. Die CCCHs werden von mehreren MS's verwendet. [TS-04.03, Kap. 4.1.2]

- **RACH**: Auf dem einzigen Kanal den es nur im Uplink gibt, können MS's mit einem sogenannten „Channel Request" vom Netzwerk einen dedizierten Kanal anfordern.

- **AGCH**: Der AGCH wird verwendet, um einer MS nach einem Channel-Request einen dedizierten Kanal zuzuweisen.

- **PCH**: Auf dem PCH werden MS's im Empfangsbereich einer Basisstation über eingehende Anrufe und SMS informiert.

Die bidirektionalen **Dedicated Control Channels (DCCHs)** sind einer dedizierten Verbindung zwischen BTS und MS zugeordnet. Um sie vor Zugriff von anderen zu schützen wird der Datentransfer auf DCCHs in der Regel verschlüsselt. [TS-04.03, Kap. 4.1.3]

- **FACCH**: Der FACCH ist ein Signalisierungskanal, der auf einem TCH übertragen wird. Er wird für dringende Signalisierungsnachrichten verwendet, wobei die Nutzdaten des Bursts durch Signalisierungsinformationen ersetzt werden. Ob der TCH-Burst vom FACCH gestohlen wurde, wird durch Setzen eines „Stealing Flags" gekennzeichnet. Der Qualitätsverlust der Sprachübertragung durch die fehlenden Nutzdaten ist nicht hörbar.

- **SACCH**: Der SACCH wird im Uplink für die Übertragung von Messdaten, wie der Signalstärke benachbarter BTS, verwendet. Im Downlink erhält die MS Befehle zu Leistungsregelung von der BTS.

- **SDCCH**: Der SDCCH wird für die Signalisierung bei der Nutzung von Netzdiensten verwendet, zum Beispiel für den Aufbau eines Gesprächs, das Einrichten einer verschlüsselten Verbindung, Identitätsabfragen, Authentifizierung und „Location Updates“. Freie Kapazitäten werden für den Empfang und Versand von SMS Nachrichten benutzt. In diesem Fall dient er auch der Übertragung von Nutzdaten.

Der **TCH** ist ein bidirektionaler, dedizierter Nutzdatenkanal, über den entweder Sprachdaten oder leitungsvermittelte Datendienste übertragen werden.

3.3.2. Service Access Point

Ein Service Access Point (SAP) entspricht ist eine interne Schnittstelle zwischen Protokollschichten. Die Kommunikation zwischen den Schichten basiert dabei, wie in GSM üblich, auf Primitives. Es gibt folgende Typen von Primitives [TS-04.04]:

- **Indication (IND)**: Layer 1 → Layer 2+
  Benachrichtigung höherer Schichten über Vorkommnisse auf dieser Schicht – wie zum Beispiel eingehende Daten.

- **Request (REQ)**: Layer 1 ← Layer 2+
  Höhere Schichten können damit vom SAP angebotene Dienste nutzen.

- **Subscriber Response in GSM Authentifizierung (RES)**: Layer 1 ← Layer 2+
  Bestätigung des Empfangs einer Indication.

- **Confirm (CON)**: Layer 1 → Layer 2+
  Antwort auf Response nach erfolgreich abgearbeiteter Routine.

Folgende Abbildung aus TS-04.04 zeigt die Schnittstellen der physikalischen Schicht in GSM zu anderen Protokollschichten. Die physikalische Schicht bietet SAP-Schnittstellen zur Datensicherungsschicht und dem Schicht 3 zugeordneten Radio Resource (RR)-Management, die über PH-Primitives und MPH-Primitives angesprochen werden können.

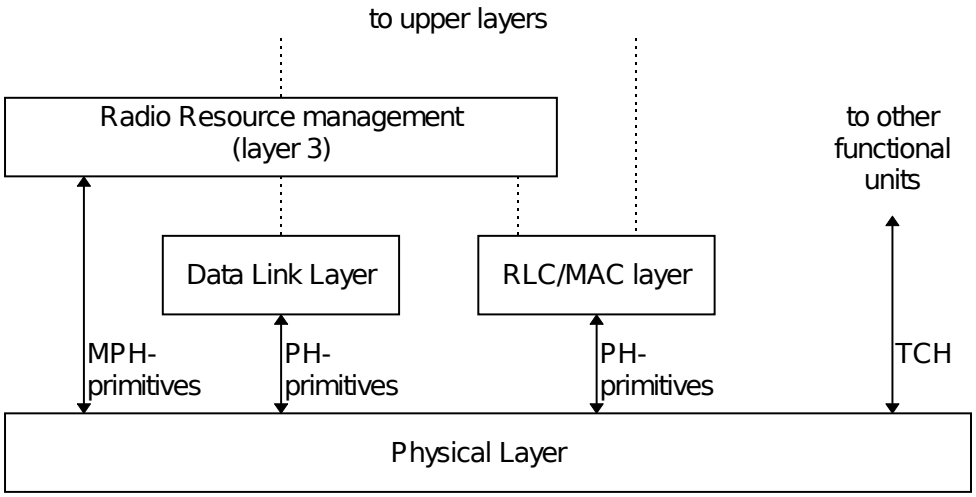


Abbildung 3.5.: Die Schnittstellen der physikalischen Schicht, aus [TS-04.04, Abb. 2.1]

Die logischen Kanäle definieren im Fall der physikalischen Schicht unterschiedliche SAPs zu LAPDm, wie in Abbildung 3.6 zu sehen ist.

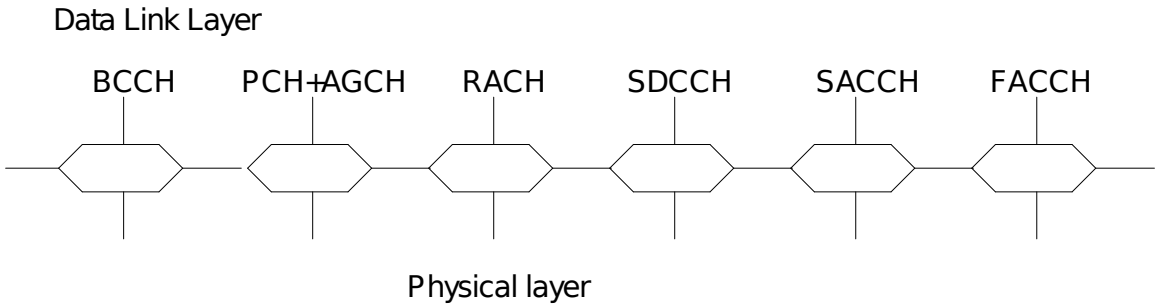


Abbildung 3.6.: SAPs der physikalischen Schicht zur Sicherungsschicht, aus [TS-04.04, Abb. 3.2]

3.3.3. Link Access Protocol Dm-Channel

Die Aufgabe des in TS-04.05 und TS-04.06 spezifizierten LAPDm-Protokolls der Datensicherungsschicht ist der Aufbau einer zuverlässigen Verbindung zwischen MS und BTS. Für die zuverlässige Übertragung von Daten stellt es einen SAP für Schicht 3 zur Verfügung, selbst greift es auf die von der physikalischen Schicht angebotenen SAPs der logischen Kanäle zu (siehe Abbildung 3.6). Das zwischen MS und BTS terminierende Protokoll wurde in großen Teilen aus der Integrated Services Digital Network (ISDN)-Spezifikation

des Protokolls Link Access Procedure for the D-Channel (LAPD) übernommen und an die Anforderungen der Funkschnittstelle angepasst.

Das Protokoll soll mehrere Einheiten von physikalischer Schicht und Schicht 3 unterstützen, sowie die Signalisierung auf BCCH, CCCH und DCCH. In TS-04.05, Kap. 3.1 wird sein Funktionsumfang wie folgt beschrieben:

- Unterstützung einer oder mehrerer Schicht 3 Datenverbindungen, die durch einen Data Link Connection Identifier (DLCI) identifiziert werden.
- Unterscheidung verschiedener Nachrichtentypen.
- Transparente Nachrichtenübertragung zwischen Schicht 3 Einheiten.
- Sicherstellen der korrekten Reihenfolge der Nachrichten (Ablaufsteuerung).
- Anpassen der Datenrate an Eigenschaften der physikalischen Schicht (Flusskontrolle).
- Benachrichtigung von Schicht 3 Einheiten über nicht behebbare Fehler.

LAPDm bietet Schicht 3 zwei verschiedenen Übertragungsoperationen an. Die unzuverlässige „unacknowledged operation“ und die zuverlässige „acknowledged operation“. Auf BCCH und CCCH ist nur die unzuverlässige Operation verfügbar, auf DCCH können beide benutzt werden. Mit den beiden Übertragungstypen realisiert LAPDm die Dienste „Unacknowledged Information Transfer“ für unzuverlässige und „Acknowledged Multiple Frame Information Transfer“ für zuverlässige Datenverbindungen.

Unacknowledged Information Transfer

Von Schicht 2 wird weder die korrekte Reihenfolge verifiziert, noch dass eine gesendete Nachricht beim Empfänger angekommen ist [TS-04.05, Kap. 4.2.4]. Nachrichten als Unumbered Information (UI)-Frames übertragen.

Acknowledged Multiple Frame Information Transfer

Durch Kontrollmechanismen von Schicht 2 wird sichergestellt, dass übertragene Daten in der richtigen Reihenfolge beim Empfänger ankommen [TS-04.05, Kap. 4.2.5]. Auf beiden Seiten werden jeweils die Nummern N(R) empfangener und N(S) gesendeter Nachrichten gepflegt. Die übertragenen Numbered Information Transfer Format (I)-Frames enthalten die beiden Nummern. Durch Synchronisation der empfangenen Nummern mit den eigenen kann so festgestellt werden, ob Daten verloren gegangen sind oder die Reihenfolge nicht mehr stimmt. Mit einer Reject (REJ) Nachricht kann eine fehlerhaft empfangene Nachricht abgewiesen werden, mit Receive-Ready (RR)-Nachrichten wird der Gegenseite mitgeteilt, welche Nachricht sie als nächstes schicken soll. Ein RR enthält dazu die Nummer $N(R) + 1$ und bestätigt gleichzeitig den Empfang aller Nachrichten bis zu dieser. Sollte die Datensicherungsschicht keine Möglichkeit haben eine zuverlässige Verbindung zu gewährleisten, wird ein Fehler an Schicht 3 gemeldet. Mit einer Set Asynchronous Balanced Mode (SABM)-Nachricht signalisiert Schicht 2 ihrer Gegenseite, dass in den zuverlässigen Übertragungsmodus gewechselt werden soll. Diese bestätigt den Wechsel mit einer Unnumbered Acknowledgement (UA)-Nachricht. Mit einem Disconnect (DISC) kann von beiden Seiten die Verbindung wieder beendet und in den unzuverlässigen Modus gewechselt werden [TS-04.06, Kap. 5.4].

Aufbau eines LAPDm-Frames

Für LAPDm-Nachrichten auf verschiedenen logischen Kanälen werden auch verschiedene Formate definiert. Da für die Masterarbeit nur LAPDm Nachrichten auf DCCH untersucht werden, wird hier nur auf das „Frame Format A“ eingegangen. Dieses ist, wie auch die anderen Formate, in TS-04.06, Kap. 2 definiert.

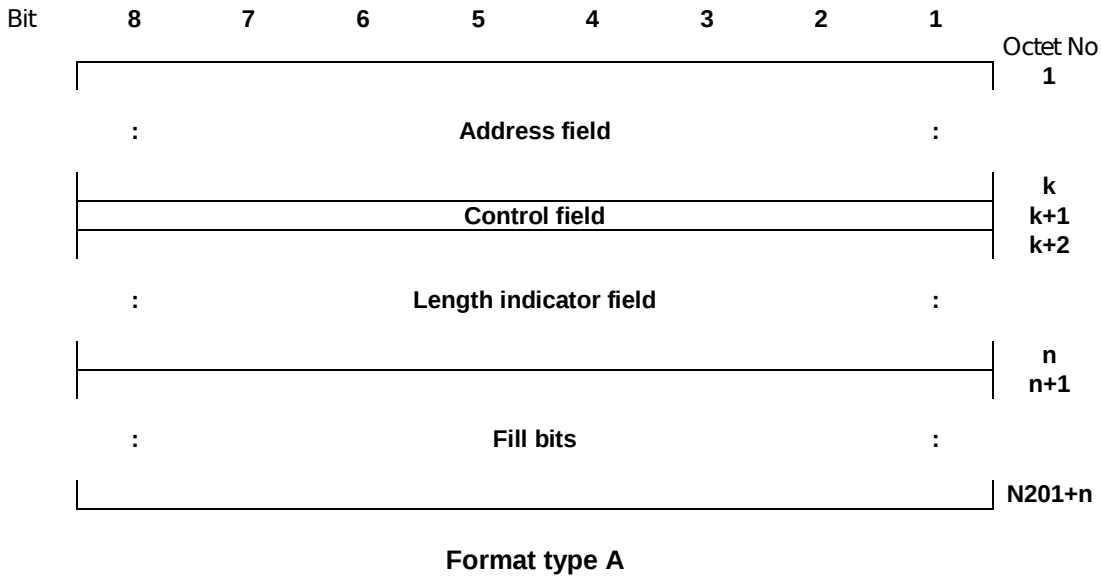


Abbildung 3.7.: Das LAPDm Format A, aus [TS-04.06, 2008, Abb. 1, Teil 1]

Obige Abbildung zeigt die LAPDm-Nachricht im Format A. Sie setzt sich aus je einem Byte (Oktett) für Adressierungs-, Kontroll- und Längeninformationen und N201 Bytes an Nutzdaten zusammen. die Größe von N201 wird in TS-04.06, Kap. 5.8.3 für FACCH und SDCCH als 20 definiert, damit ist die gesamte Nachricht 23 Bytes oder 184 Bit lang.

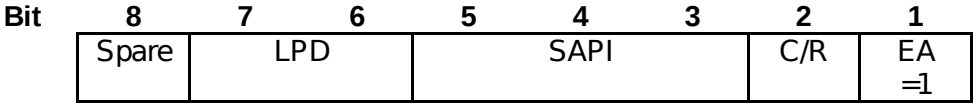


Abbildung 3.8.: Das LAPDm Addressierungsfeld, aus [TS-04.06, 2008, Abb. 4]

Das oben gezeigte Adressierungsfeld liefert Informationen über den Sender und Empfänger der Nachricht. Das End of Address Bit (EA) sagt aus, ob dies das letzte Byte des Adressierungsfeldes ist oder noch weitere folgen. Es wird benötigt, da das Adressierungsfeld auch mehrere Bytes lang sein kann. Das Command/Response Bit (C/R) definiert ob es sich um einen Befehl (Command) oder die Antwort auf diesen (Response) handelt. Die SAPI entspricht dem DLCI und definiert die Schicht 3 Datenverbindung, für die die Nachricht bestimmt ist. Das Link Protocol Discriminator (LPD)-Feld identifiziert das verwendete Schicht 2 Protokoll und ist in LAPDm immer 0.

Control field bits	8	7	6	5	4	3	2	1
I format	N(R)			P	N(S)			0
S format	N(R)			P/F	S	S	0	1
U format	U	U	U	P/F	U	U	1	1

Abbildung 3.9.: Das LAPDm Kontrollfeld, aus [TS-04.06, 2008, Tabelle 3]

Der oben gezeigte Aufbau des Kontrollfelds hängt mit dem verwendeten Nachrichtenformat zusammen. Beim nummerierten I-Format (Bit[1] == 0) gibt es die Nummer der letzten korrekt empfangenen Nachricht N(R) und die Nummer der zuletzt gesendeten Nachricht N(S). Das Supervisory Format (S) (Bit[1,2] == 10) benötigt die Nummer N(R) der erwarteten nächsten Nachricht und zwei S-Bits, die die verwendete Supervisory Funktion (zum Beispiel RR) bestimmen. Das Unnumbered Information Transfer Format (U) (Bit[1,2] == 11) ist nicht durchnummeriert und benutzt alle U-Bits für die Bestimmung der verwendeten Unnumbered Funktion (zum Beispiel SABM). Das Poll/Final Bit (P/F) definiert bei einem Command, ob man eine Antwort der Gegenseite möchte und bei einer Response, ob es sich um eine Antwort auf einen „Poll“ der Gegenseite handelt.

Bit	8	7	6	5	4	3	2	1
	L						M	EL =1

Abbildung 3.10.: Das LAPDm Längeninformationsfeld, aus [TS-04.06, 2008, Abb. 5]

Das End of Length Bit (EL) der oben gezeigten Längeninformation sagt aus, ob es sich bei diesem um das letzte Byte handelt (siehe EA im Adressierungsfeld). Das More Bit (M) zeigt an, ob sich die übertragene Signalisierungsnachricht aus mehr als dieser LAPDm-Nachricht zusammensetzt. Das wird benötigt, wenn mehr als 160 Bit an Signalisierungsinformationen übertragen werden müssen, da diese nicht mehr in eine einzelne LAPDm-Nachricht passen. Der Length Indicator (L) gibt die Länge der Nutzdaten in dieser Nachricht an. Alle Daten nach der definierten Länge enthalten keine nützlichen Information mehr und werden in der Regel mit `0x2b` aufgefüllt.

3.4. Schicht 3 Protokolle

Die Protokollebene 3 besteht in GSM aus Radio Resource (RR), Mobility Management (MM) und Connection Management (CM), die verschiedene Aufgaben übernehmen. Ein

Teil des RR-Protokolls fällt in den Aufgabenbereich der BTS und wird von dieser bearbeitet, der Rest wird transparent an den BSC weitergeleitet. MM und CM werden weder von BTS noch BSC bearbeitet und den Protokollen zugewiesene Nachrichten zwischen MS und MSC ausgetauscht. Der für die Funkschnittstelle relevante Teil der Protokolle ist in TS-04.18 spezifiziert, der Rest findet sich in TS-24.008 und TS-23.108.

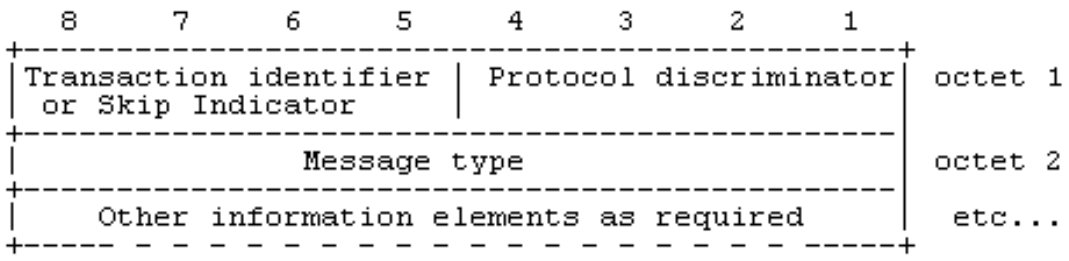


Abbildung 3.11.: Der gemeinsame Header der Schicht 3 Protokolle, aus [TS-24.008, 2005, Abb. 10.1]

Die Protokolle teilen sich den in Abbildung 3.11 gezeigten, gemeinsamen Header [TS-24.007, 2005, 11.2.3.1]. Bit 1 bis 4 werden einem Protokolldiskriminator zugeordnet und ermöglichen die Unterscheidung der verschiedenen Schicht 3 Protokolle. Die für die Arbeit relevanten Werte für diesen sind in Tabelle 3.2 aufgeführt. Bit 5 bis 8 kann entweder unbenutzt bleiben und übersprungen werden („Skip Indicator“) oder für die Zuordnung der Nachricht zu einer von bis zu 16 Transaktionen oder Verbindungen (Transaction Identifier (TI)) verwendet werden. Das zweite Byte ist für den Typ der übertragenen Nachricht reserviert („Message Type“), von denen die Schicht 3 Protokolle verschiedene spezifizieren können.

Bits 4 3 2 1	Protokoll
0 0 1 1	Call Control
0 1 0 1	Mobility Management
0 1 1 0	Radio Resource Management
1 0 0 1	SMS

Tabelle 3.2.: Werte des Schicht 3 Protokolldiskriminators, nach [TS-24.007, 2005, Tabelle 11.2]

Die Informationsfelder des Schicht 3 Headers sind im Type/Length/Value (TLV) Format angegeben. Der „Type“ ist der Information Element Identifier (IEI) des Informationselements, „Length“ seine Länge und „Value“ sein Wert. Damit ist es möglich, optionale Informationselemente dynamischer Länge aneinanderzureihen. Ist die Position des Feldes

festgelegt, muss T nicht mit angegeben werden. Ist die Länge festgelegt, muss L nicht angegeben werden. Bei einem reinen V Feld ist zum Beispiel sowohl Länge als auch Typ festgelegt. Tabelle 3.3 zeigt die Kombinationsmöglichkeiten.

	T (Typ)	**L (Länge)**	**V (Wert)**	**Gesamtlänge**
V	-	-	+	fest
T	+	-	-	fest
TV	+	-	+	fest
LV	-	+	+	dynamisch
TLV	+	+	+	dynamisch

Tabelle 3.3.: Das TLV Format für Informationselemente der Protokollschicht 3, nach [TS-24.007, Kap. 11.2.1.1]

Im Folgenden wird näher auf die RR, MM und Call Control (CC)-Protokolle eingegangen.

3.4.1. Radio Resource

Radio Resource (RR) ist größtenteils für die Verwaltung der Frequenzen und Kanäle zuständig. Die Kommunikation dazu läuft zwischen den RR-Modulen von MS und BSC. Zwischen MS und BTS terminiert die Messung der Verbindungsqualität auf der Funkschnittstelle und die Regelung der Signalstärken der MS. Das Einrichten, Aufrechterhalten und Beenden von RR-Verbindungen, die eine dedizierte Verbindung zwischen MS und Netzwerk ermöglichen, ist die Hauptaufgabe von RR [TS-04.18].

Übersicht über die Aufgaben des RR im MS [TS-04.18, Kap. 3.2.1]:

- BCCH-Überwachung: Die Auswertung von SI-Nachrichten auf dem Downlink.
- PCH-Überwachung: Die Auswertung von Paging-Nachrichten auf dem Downlink.
- RACH-Verwaltung: Das Anfordern eines dedizierten Signalisierungskanals als Antwort auf Paging oder initiiert vom MS.
- Der Aufbau von RR-Verbindungen auf dedizierten Kanälen.

- Der Transport von Nachrichten über RR-Verbindungen.
- Das Aushandeln von Verschlüsselungsalgorithmen und die Einrichtung verschlüsselter Verbindungen.
- Die Messung und Mitteilung der Verbindungsqualität an die BTS.

Übersicht über die Aufgaben des RR im Netzwerk [TS-04.18, Kap. 3.2.2]:

- Die Zuweisung und der Aufbau von RR-Verbindungen auf dedizierten Kanälen.
- Der Transport von Nachrichten über RR-Verbindungen.
- Das Aushandeln von Verschlüsselungsalgorithmen und die Einrichtung verschlüsselter Verbindungen.
- Die Überwachung der Verbindungsqualität der MS und die Regelung von deren Sendeleistung.
- Die Durchführung von Handover-Prozeduren, also die Übergabe von laufenden Gesprächen zwischen BSC oder BTS, wenn sich ein MS in einen neuen Zuständigkeitsbereich begibt.

3.4.2. Mobility Management

Mobility Management (MM) ist für alle Funktionen und Abläufe zuständig, die sich aus der Mobilität des Mobilfunkteilnehmers ergeben. Die komplette MM-Signalisierung läuft transparent für das BSS, zwischen MSC und MS, ab. Für die Nachrichtenübertragung greift das MM über einen SAP auf die vom RR zur Verfügung gestellte, dedizierte Verbindung zu. Das Protokoll ist in TS-24.008 spezifiziert, Beispiele für die Abläufe finden sich in TS-23.108.

Übersicht über die Aufgaben des MM [TS-24.008, Kap. 4]:

- Die Zuweisung von TMSIs zum Schutz der Teilnehmeridentitäten.

- Die Lokalisierung der MS im BSS, also die Bearbeitung von „Location Updates“.
- Die Zuordnung von IMSIs und International Mobile Equipment Identitys (IMEIs) zu MS.
- Die Authentifizierung von Netzteilnehmern.
- Die Registrierung („IMSI-Attach“) und Abmeldung („IMSI-Detach“) von Netzteilnehmern beim Einlegen und Entfernen der SIM-Karte.

3.4.3. Connection Management

Connection Management (CM) ist für den Aufbau von Verbindungen zwischen Endgeräten zuständig, dazu gehören Telefonate und der Versand von SMS. CM wird noch einmal unterteilt in Call Control (CC), Supplementary Services (SS) und SMS. Ebenso wie beim MM sind Nachrichten des CM transparent für das BSS [TS-24.007].

Übersicht über die Aufgaben des CM:

- CC kümmert sich um den Aufbau, die Verwaltung und das Beenden von Anrufen [TS-24.008, Kap. 5].
- SS kann anrufbezogen sein oder nicht. Der anrufbezogene Teil ist zuständig für Anrufweiterleitung, Halten und Warten, sowie Gruppenanrufe [TS-24.010].
- SMS regelt den Empfang und Versand von „Point-to-Point“ Kurznachrichten [TS-24.011].

3.5. Signalverarbeitung und Kanalkodierung

Die im Digital Signal Processor (DSP) meist in Hardware implementierte Logik für digitale Signalverarbeitung, Kodierung und Verschlüsselung von Daten wird der physikalischen Schicht zugeordnet. Abbildung 3.12 zeigt, welche Schritte Sprachdaten und sonstige digitale Daten durchlaufen, bevor sie als analoges Signal auf der Trägerfrequenz gesendet werden.

Im Gegensatz zu anderen digitalen Daten durchläuft Sprache noch einen zusätzlichen Kodierungsschritt. Sprachkodierungen wie Full-Rate-, Half-Rate-, Enhanced-Full-Rate und Adaptive-Multi-Rate-Speech-Codec sorgen für eine verlustbehaftete Kompression der Sprachdaten, um die benötigte Bandbreite zu reduzieren. [Sauter, 2011, S. 55 ff.].

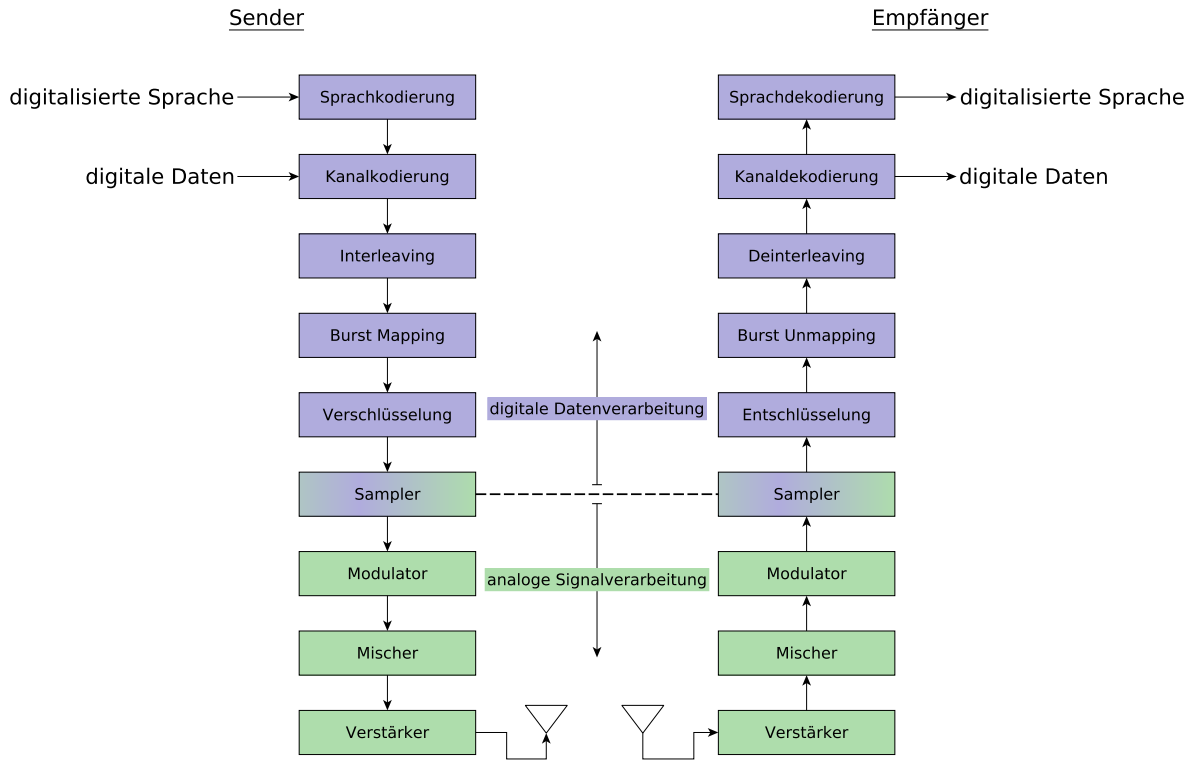


Abbildung 3.12.: Die Signalverarbeitung in GSM, erstellt mit yEd nach [TS-05.03, Figure 1a] und [Zou]

Beim Sendevorgang laufen die diskreten, digitalen Daten nach Kanalkodierung und Verschlüsselung durch den Sampler, der sie in ein analoges Signal umwandelt. Dieses wird dann auf die bestehende Trägerfrequenz moduliert, gemischt und verstärkt übertragen. Beim Empfang der Daten werden die Schritte der digitalen und analogen Signalverarbeitung umgekehrt durchlaufen [Zou].

Aufgrund der hohen Anfälligkeit der Signalübertragung über die Luft werden Mechanismen zur Fehlererkennung und Korrektur angewandt, mit denen die Bitfehlerrate um den Faktor 10^3 verringert werden kann [Eberspächer u. a., 2008, Kap. 4.8]. Jeder logische Kanal hat andere Anforderungen an die übertragenen Daten. Deshalb werden, wie in Abbildung 3.13 zu sehen, verschiedene Kodierungsverfahren kombiniert.

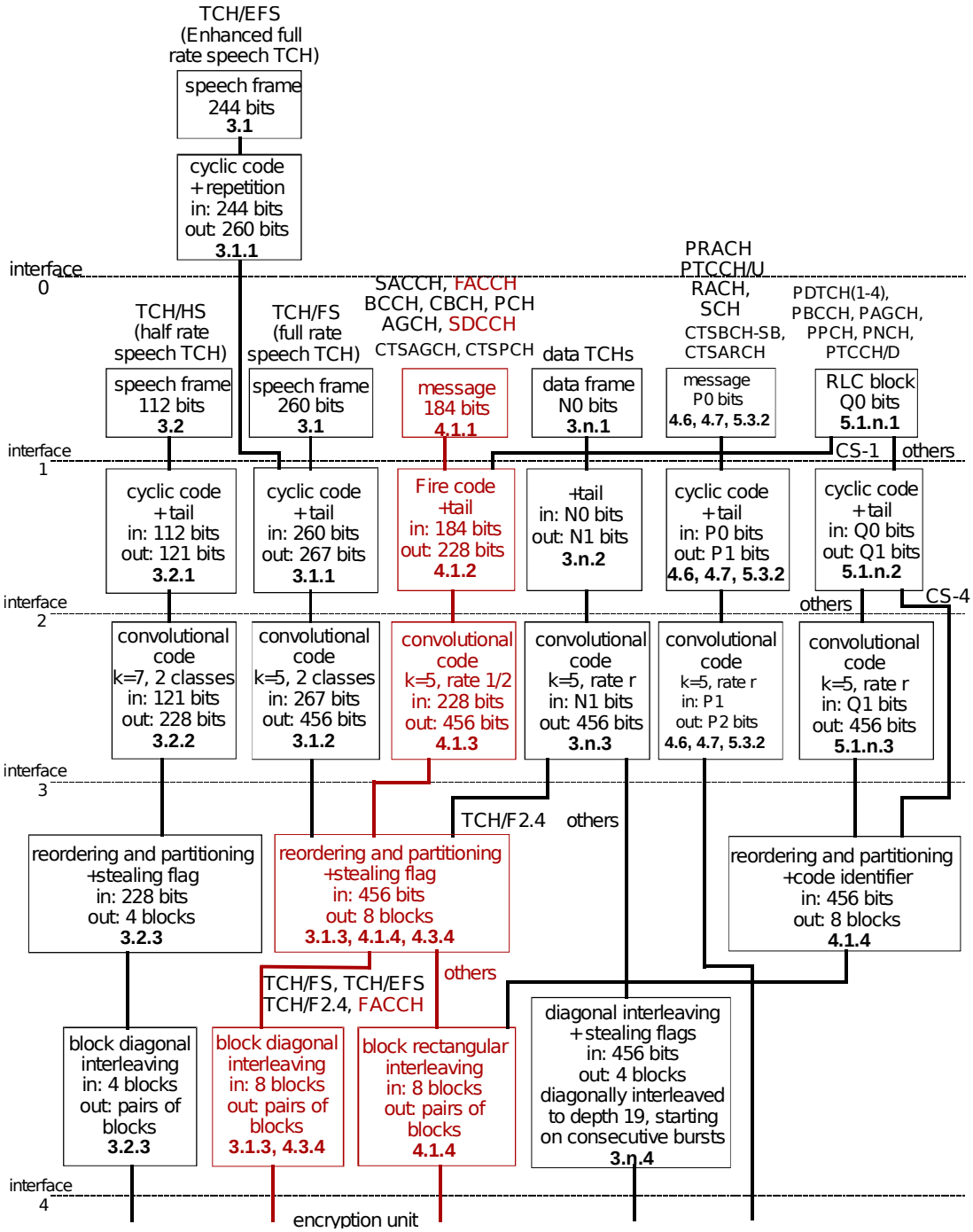


Abbildung 3.13.: Die Kanalkodierung in GSM, aus [TS-05.03, Abb. 1a]

In Abbildung 3.13 aus TS-05.03 [2005] sind die Kodierungsschritte aufgelistet, die für die verschiedenen logischen Kanäle durchlaufen werden. In jedem Kasten findet man unter dem

Namen des Verfahrens die Nummer des Kapitels aus TS-05.03 [2005], in dem es beschrieben wird. Die Call Setup Nachricht, die für den vorgestellten Angriff manipuliert werden muss, wird auf den Kanälen FACCH oder SDCCH übertragen. Die farblich markierten Verfahren sind also für die Arbeit besonders relevant und werden im Folgenden genauer erklärt.

3.5.1. Blockcode

Mit Blockcodes wird einem Block von Daten Redundanz für Fehlererkennung und/oder Fehlerkorrektur hinzugefügt. Der GSM-Standard definiert den Einsatz von zwei Verfahren, Cyclic Redundancy Check (CRC) für Nutzdatenkanäle und Firecode auf Signalisierungskanälen.

Auf Sprachkanälen werden die Informationsbits zudem in verschiedene Klassen eingeteilt, von denen nur Klasse 1 Bits besonders signifikant für die Spracherkennung sind und durch den CRC fehlergeschützt werden. Klasse 2 Bits sind nicht so wichtig und fließen deshalb auch nicht in die Berechnung der Paritätsbits mit ein.

Für Signalisierungsnachrichten wird ein Firecode verwendet, ein linearer binärer Blockcode. Für eine Reihe von Informationsbits liefert dieser eine Anzahl redundanter Bits, die für die Fehlererkennung und -korrektur verwendet werden können. Die redundanten Bits werden durch exklusives oder (XOR), beziehungsweise binäre Addition der Informationsbits berechnet.

Für alle zyklischen Codes kann die Berechnung der Redundanz als binäre Polynomdivision ausgedrückt werden. Aus der originalen Datensequenz wird ein Polynom aufgebaut, dessen Koeffizienten die Bits der Sequenz sind. Der Teiler ist ein definiertes Generatorpolynom und die Redundanz der Rest der Polynomdivision. Durch das Anhängen der Redundanz an die Datensequenz kann der Empfänger diese auf Fehler überprüfen, indem er ebenfalls durch das Generatorpolynom teilt. Ist das Ergebnis 0, sind keine Fehler aufgetreten. Es ist möglich, einen Sollrest anzugeben, der bei der Überprüfung der Datensequenz als Ergebnis herauskommen soll. Durch die XOR-Verknüpfung der berechneten Redundanz mit dem Sollrest auf der Senderseite kommt dieser bei der Überprüfung durch den Empfänger wieder heraus. GSM definiert als Sollrest die 40 Bits mit dem Wert 1, wodurch die Daten

als fehlerfrei übertragen gelten, wenn das Ergebnis der Überprüfung `0xfffffffffff` ist.

$$(x^{23}+1)\cdot(x^{17}+x^3+1) = x^{40}+x^{26}+x^{23}+x^{17}+x^3+1 \tag{3.1}$$

Das Generatorpolynom (siehe Gleichung 3.1) des Firecodes und die Definition des Soll-Rests ist in TS-05.03, Kap. 4.1.2 zu finden.

Da die Eingangsdaten immer ein LAPDm Frame sind, ist die Länge k der Informationsbits 184 Bit. Der Firecode generiert mit obigem Generatorpolynom, entsprechend dessen Grad, aus der 184 Bit langen Sequenz, 40 redundante Paritätsbits $p(k)$. In GSM werden die berechneten Paritätsbits einfach an die Originaldaten $d(k)$ angehängt. Für die Weiterverarbeitung werden für die Faltungskodierung noch vier Nullbits angefügt, womit die ausgehenden Daten $u(k)$ die Länge von 228 Bit haben. Der Zusammenhang ist in Gleichung 3.2 mathematisch dargestellt.

$$\begin{aligned} u(k) &= d(k) && \text{für } k = 0, 1, ..., 183 \\ u(k) &= p(k-184) && \text{für } k = 184, 185, ..., 223 \\ u(k) &= 0 && \text{für } k = 224, 225, 226, 227 \end{aligned} \tag{3.2}$$

Das verwendete Generatorpolynom wurde von Fire [1959] vorgestellt und bietet eine gute Erkennung und Korrektur von Fehlergruppen von bis zu 11 Bits. Obwohl eine Fehlerkorrektur möglich wäre, verzichtet GSM darauf und verlässt sich stattdessen auf die erneute Übertragung der Nachricht, auf der von LAPDm zur Verfügung gestellten, zuverlässigen Verbindung. In folgendem Beispiel wir der Firecode auf Testdaten angewendet.

```
Datensequenz (184 Bit):
01 20 51 03 45 04 04 60  02 00 81 5e 07 81 10 57
81 81 81 81 15 01 01

Rest aus Polynomdivision mit Generatorpolynom (40 Bit):
07 47 e3 10 1a

Soll-Rest in GSM (40 Bit):
ff ff ff ff ff

Paritätsbits in GSM == Rest XOR Soll-Rest (40 Bit):
f8 b8 1c ef e5
```

```
Anhängen von Paritäts- und Nullbits (228 Bit):
01 20 51 03 45 04 04 60  02 00 81 5e 07 81 10 57
81 81 81 81 15 01 01 f8  b8 1c ef e5 0
```

Codebeispiel 3.1: Kodierung von Testdaten mit dem Firecodes, Datensatz generiert mit `dummycoder` (siehe Abschnitt 6.4)

Blockcodes und zyklische Codes gehören zu den linearen Codes, weshalb Methoden der Linearen Algebra angewandt werden können. Für lineare Abbildungen gilt Additivität [Werner, 2008, S. 142 ff.]:

$$\boldsymbol{u}(\boldsymbol{x} \oplus \boldsymbol{y}) = \boldsymbol{u}(\boldsymbol{x}) \oplus \boldsymbol{u}(\boldsymbol{y}) \tag{3.3}$$

Es können also die Paritätsbits berechnet werden, die bei einer Datenmanipulation kippen. Das bedeutet Blockcodes, wie der Firecode, sind nur für die Erkennung von zufälligen Fehlern ausgelegt und eignen sich nicht für den Schutz der Integrität.

3.5.2. Faltungscode

Durch Faltungskodierung, auch „Convolutional Coding“ genannt, wird das Signal erneut mit Redundanz zur Fehlerkorrektur versehen. Wie Blockcodes lässt sich auch die Faltungskodierung polynomial beschreiben. Bei der Kodierung wird der Informationsgehalt der Eingangsbits, durch Faltung mit einer durch das Generatorpolynom definierten Maske, auf mehrere Ausgangsbits verteilt. Bei der Dekodierung kommt der Viterbi-Algorithmus zum Einsatz, der aus der kodierten Datensequenz die wahrscheinlichsten Ausgangsdaten bestimmt. Da bei der Dekodierung die bereits durch den Viterbi-Dekodierer fehlerkorrigierten Daten in den Paritätscheck eingehen, spricht man bei der Faltungskodierung von internem und beim Blockcode von externem Fehlerschutz [Eberspächer u. a., 2008, Kap. 4.8.1, 4.8.2].

Die Polynomdarstellung des in GSM verwendeten „1/2 Rate Convolutional Coders“ ist in TS-05.03, Kap. 4.1.3 zu finden.

$$\begin{aligned} g_0 &= x^4 + x^3 + 1 \\ g_1 &= x^4 + x^3 + x + 1 \end{aligned} \tag{3.4}$$

Obige Gleichung bedeutet, das aus einem Eingangsbit x zwei Bits g_0 und g_1 durch binäre

Additionen von x mit seinen Vorgängerbits generiert werden. Wo es keine Vorgängerbits gibt, werden diese als 0 definiert.

Die Eingangsdaten der Faltungskodierung sind die vom Blockcode generierte Datensequenzen. Das Ergebnis $c(k)$ der Faltungskodierung wird wie folgt als Funktion der Eingangsdaten $u(k)$ ausgedrückt:

$$\begin{aligned} c(2k) &= u(k-4) \oplus u(k-3) \oplus u(k) \\ c(2k+1) &= u(k-4) \oplus u(k-3) \oplus u(k-1) \oplus u(k) \quad \text{für } k = 0, 1, ..., 227 \\ u(j) &= 0 \quad \forall\, j < 0 \end{aligned} \tag{3.5}$$

Die Faltungskodierung wird in Hardware als Schieberegister implementiert. Es wird ein Puffer von vier Bits benötigt, da die Bits bis zu vier Stellen vor dem Eingangsbit ($u(k-4)$) in die Berechnung mit eingehen. Der Puffer wird mit vier Speicherregistern realisiert. Durch die mit XOR-Gattern implementierten Funktionen $c(2k)$ und $c(2k+1)$ werden zwei Ausgangsbits generiert. Welche Speicherregister verknüpft werden müssen, lässt sich aus Gleichung 3.5 ableiten. Da pro Eingangsbit zwei Ergebnisbits generiert werden, bezeichnet man den Kodierer als $1/2$ Rate Convolutional Coder. Die vier nach dem Blockcode angehängten Nullbits sind notwendig, um das Schieberegister wieder in einen vordefinierten Zustand – alle Register gleich 0 – für den nächsten Block zu bringen.

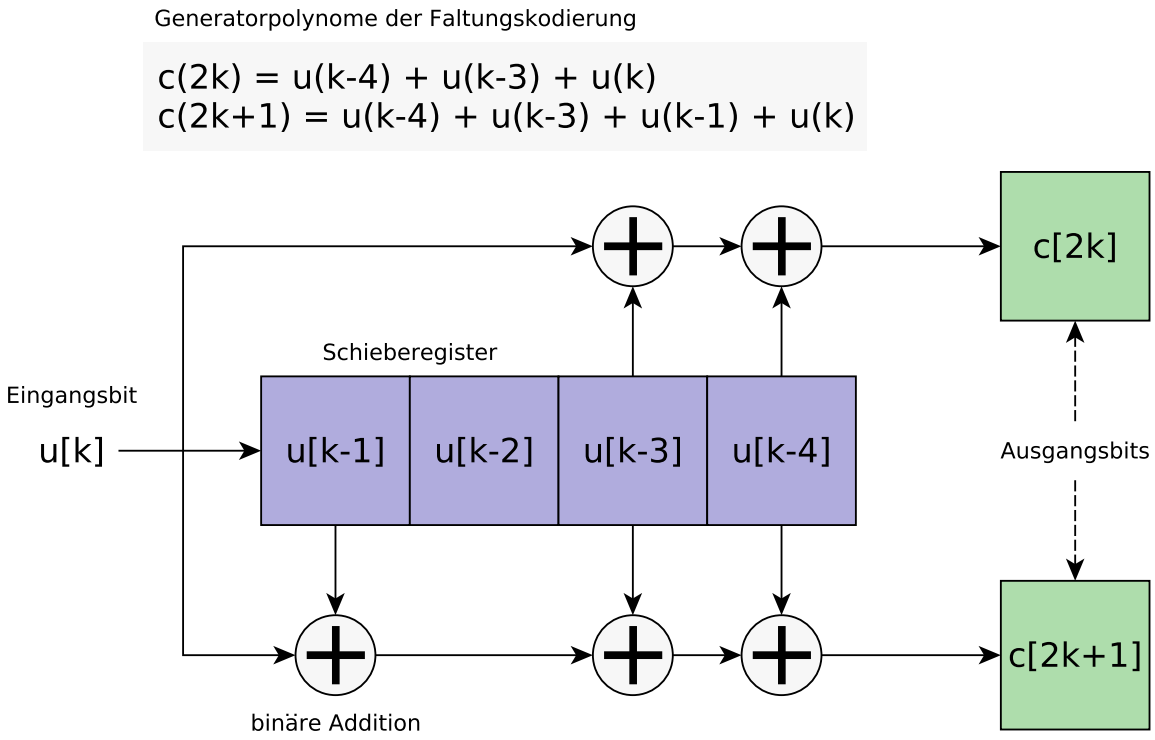


Abbildung 3.14.: Das Schieberegister für die Faltungskodierung, erstellt mit yEd

Auf Signalisierungskanälen sind alle Datenbits wichtig und fließen in die Faltungskodierung mit ein. Bei Sprachkanälen hingegen wird ein sogenannter „Punctured Convolutional Code“ verwendet. Um Bandbreite zu sparen werden dabei vom Kodierer nur die für die Wiederherstellung des Sprachsignals wichtigen Klasse 1 Bits kodiert, die unwichtigeren Klasse 2 Bits werden ohne Redundanz übertragen [TS-05.03, Kap. 3.1.2, 3.2.2].

Wie Blockcodes gehört die Faltungskodierung zur Gruppe linearer Codes und es gilt Additivität [Werner, 2008, S. 142 ff.]:

$$\boldsymbol{c}(\boldsymbol{x} \oplus \boldsymbol{y}) = \boldsymbol{c}(\boldsymbol{x}) \oplus \boldsymbol{c}(\boldsymbol{y}) \tag{3.6}$$

In folgendem Beispiel wir die Faltungskodierung auf Testdaten angewendet.

```
Datensequenz (228 Bit):
01 20 51 03 45 04 04 60  02 00 81 5e 07 81 10 57
81 81 81 81 15 01 01 f8  b8 1c ef e5 0
```

```
Faltungskodierte Daten (456 Bit):
00 03 42 3c 37 bc 4f 0e  47 c7 bf 34 f0 34 c9 cc
00 0d 3c 00 d3 c3 78 55  0c 3a 50 c3 4c 4f 37 85
50 c3 9c c3 9c c3 9c c3  4c 78 bf 03 4f 03 a6 90
1d 60 c3 a8 da e5 a9 3b  bf
```

Codebeispiel 3.2: Faltungskodierung von Testdaten, Datensatz generiert mit `dummycoder` (siehe Abschnitt 6.4)

3.5.3. Interleaving

Um gegen auf der Funkschnittstelle auftretende Burstfehler zu schützen, werden die Daten umsortiert und vom Interleaver verschachtelt. Blockweise Signalstörungen verteilen sich damit auf eine größere Datenmenge und können von der Fehlerkorrektur mit größerer Wahrscheinlichkeit berichtigt werden. Die aus dem Faltungskodierer kommenden Datenblöcke N_n mit 456 kodierten und verschlüsselten Bits werden in Unterblöcke von je 57 Bits unterteilt. Je zwei solcher Unterblöcke werden einem Burst B_b zugewiesen (siehe Abbildung 3.3). Für Sprachkanäle wird die Stelle im Burst $B_b[j]$, an dem ein Bit eines Datenblocks $N_n[k]$ landet, in TS-05.03, Kap. 3.1.3 wie folgt bestimmt:

$$
\begin{aligned}
&i(B_b, j) = c(N_n, k) \\
&k = 0, 1, ..., 455 \\
&n = 0, 1, ... \\
&b = 4 \cdot n + (k \bmod 8) \\
&j = 2 \cdot ((49 \cdot k) \bmod 57) + ((k \bmod 8) \operatorname{div} 4)
\end{aligned}
\tag{3.7}
$$

Die Nummer des Bursts wird also durch die Nummer des Datenblocks und den Laufindex k modulo 8 auf diesem bestimmt. Das heißt jedes der acht Bits eines Bytes wird in einem anderen Burst untergebracht. Wird ein kompletter Burst fehlerhaft übertragen, erhält man auf Empfängerseite nach dem Deinterleaving die 114 fehlerhaften Bits auf einen Fehler pro Byte verteilt. So verteilte Bitfehler können von der Fehlerkorrektur berichtigt werden.

In TS-05.03, Tabelle 1 ist das Ergebnis von Interleaving und Umsortierung der Bits aufgelistet. Die Daten eines Blockes N_n sind so auf die Bursts verteilt, dass die vorderen vier Bursts $B_{4n+0,1,2,3}$ immer die geraden Bits [0-56] und die hinteren vier Burst $B_{4n+4,5,6,7}$

die ungeraden Bits [57-113] enthalten. Das bedeutet, dass ein Burst immer die Daten von zwei Datenblöcken N_n und N_{n+1} enthält. Die geraden Datenbits stammen aus dem höheren, die ungeraden aus dem niedrigeren Block. Diese Art der Verzahnung nennt man block-diagonales Interleaving. Alle vier Bursts beginnt ein neuer Datenblock, der in acht Bursts komplett übertragen wird. Die Übertragung einer gesamten Sprachprobe dauert somit $8 \cdot 4.615ms = 30ms$. Folgendes Beispiel zeigt das Ergebnis von block-diagonalem Interleaving, angewendet auf das das Ergebnis der Faltungskodierung aus Codebeispiel 3.2. Die Daten werden binär dargestellt und sind bereits ihren Bursts zugewiesen, zur besseren Kenntlichkeit sind binäre Einsen rot und Nullen blau markiert. Im Beispiel ist nur das Ergebnis des Interleaving für den Datenblock N_n dargestellt, nicht die Überlagerung mit vorherigem und nachfolgenden Datenblock. So kann man erkennen, dass Bit [57-113] der ersten vier Bursts und Bit [0-56] der zweiten vier Bursts nicht befüllt sind (alle Bits gleich Null). Bit [57-113] der ersten vier Bursts wäre vom Datenblock N_{n-1} befüllt worden, N_{n+1} würde Bit [0-56] der zweiten vier Bursts befüllen.

```
Bit[0-56]                                                Bit[57-113]
00000000000000100010001010100000100000101010100010001000100000000000000000000000000000000000000000000000000000000000000000 Burst 0
00101000100000101010000000000001010100010001010100010100000000000000000000000000000000000000000000000000000000000000000000 Burst 1
00101000101010100000000010101000000000001010000000010000100000000000000000000000000000000000000000000000000000000000000000 Burst 2
00000000010101000100010101000000000000010101010100000000000000000000000000000000000000000000000000000000000000000000000000 Burst 3
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 Burst 4
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 Burst 5
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 Burst 6
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 Burst 7
```

Codebeispiel 3.3: Verschachtlung von Testdaten mit block-diagonalem Interleaving, Datensatz generiert mit `dummycoder` (siehe Abschnitt 6.4)

Das für Signalisierungskanäle angewandte Interleaving [TS-05.03, Kap. 4.1.4] unterscheidet sich nur in der Verteilung der 57 Bit Blöcke auf die Bursts.

$$
\begin{aligned}
&i(B_b, j) = c(N_n, k) \\
&k = 0, 1, ..., 455 \\
&n = 0, 1, ... \\
&b = 4 \cdot n + (k \bmod 4) \\
&j = 2 \cdot ((49 \cdot k) \bmod 57) + ((k \bmod 8) \operatorname{div} 4)
\end{aligned}
\tag{3.8}
$$

Im Gegensatz zum block-diagonalen Interleaving werden die generierten Blöcke auf nur vier Bursts aufgeteilt. Es gibt keine Verzahnung von aufeinander folgenden Datenblöcken,

wodurch eine Signalisierungsnachricht nach $4 \cdot 4.615ms = 15ms$ komplett übertragen ist. Ein Burst $B_{4n+1,2,3,4}$ enthält also eine Kombination von geraden und ungeraden Daten des selben Datenblocks N_n. Das Verfahren wird deshalb block-rectangular Interleaving genannt. Die Verteilung der Bits eines Datenblocks N_n auf die Burst mit block-rectangular Interleaving, ist in folgendem Beispiel dargestellt. Es werden die selbigen Eingangsdaten verwendet wie in Codebeispiel 3.3. Anschaulich gesagt wandern Bit [57-113] der zweiten vier Bursts „nach oben“, in die ersten vier Bursts.

```
Bit[0-56]                                                 Bit[57-113]
00000000000000100010001010100000100000101010100010010001 10001010101010000000001010101000100010100000100000000000 Burst 0
00101000100000101010000000000000101010001000101010010100 10000000100000101010001010001000001010000010101000101000 Burst 1
00101000101010100000000010101000000000001010000000010001 00100000001010101000100010000010100000001010100010100000 Burst 2
00000000001010100010001010100000000000001010101010000000 10101000100000100000000010101000001000000010101010100100 Burst 3
```

Codebeispiel 3.4: Verschachtlung von Testdaten mit block-rectangular Interleaving, Datensatz generiert mit `dummycoder` (siehe Abschnitt 6.4)

3.5.4. Burst Mapping

Ein normaler Burst enthält zusätzlich zu den zwei 57 Bit Blöcken an Daten noch sogenannte „Stealing Flags“ (siehe Abbildung 3.3). Sie sind gesetzt, wenn die Daten des ihnen zugeordneten 57 Bit Blockes für die Übertragung von Signalisierungsinformationen genutzt werden. Auf Sprachkanälen kann der FACCH somit die Bandbreite des TCH mitbenutzen. Die Sprachdaten auf dem TCH, die so durch Signalisierungsdaten des FACCH ersetzt werden, werden verworfen [TS-05.03, Kap. 4.3.5]. Der FACCH stiehlt also Bandbreite vom TCH, was den Namen des Flags erklärt. Auf Kanälen wie dem SDCCH, die nur Signalisierungsinformationen übertragen, sind immer beide Flags gesetzt [TS-05.03, Kap. 4.5, 4.1.5].

„Burst Mapping“ fügt in die vom Interleaving erstellten 57 Bit Blöcke Stealing Flags ein und ordnet jeweils zwei der entstandenen 58 Bit Blöcke den 116 Bit Nutzdaten eines fertigen Bursts zu. Das Beispiel zeigt die Zuordnung der schon in Unterabschnitt 3.5.3 verwendeten, auf einem SDCCH übertragenen Signalisierungsnachricht. In diesem Fall werden beide Stealing Flags gesetzt. Würde die Signalisierungsnachricht auf einem FACCH übertragen werden, wäre sie vom Burstmapping auf acht Bursts verteilt worden. Jedem der acht Bursts würde dann nur die Hälfte seiner Nutzdaten gestohlen werden.

```
Bit[0-56]                                                 Bit[59-115]
00000000000000100010001010100000100000101010100010001 1 1 10001010101010000000001010101000100010100000100000000000 Burst 0
00101000100000101010000000000000101010001000101010001010 1 1 10000000100000101010001010001000001010000010101000101000 Burst 1
00101000101010100000000010101000000000000101000000000010001 1 1 00100000001010101000100010000010100000001010100010100000 Burst 2
00000000001010100010001010100000000000001010101010000000 1 1 10101000100000100000000010101000001000000010101010100100 Burst 3
                                                          ^ ^
                Stealing Bit [57] für Daten von Bit[0-56]  |  Stealing Bit [58] für Daten von Bit[59-115]
                                  <-----------------  |  ----------------->
```

Codebeispiel 3.5: Zuordnung von Testdaten zu Bursts auf dem SDCCH durch Burstmapping, Datensatz generiert mit `dummycoder` (siehe Abschnitt 6.4)

3.6. Sicherheit in GSM

Da auf der Funkschnittstelle jeder Mithören kann, der ein entsprechendes Empfangsgerät besitzt, spezifiziert GSM verschiedene Techniken und Funktionen, um den Schutz, sowie die Authentizität der Identitäten der Netzteilnehmer und die Vertraulichkeit der kommunizierten Daten zu gewährleisten.

Die Teilnehmeridentität wird durch Verwendung einer TMSI zur Identifikation des MS beim Netzwerk geschützt. Ein Teilnehmer erhält vom Netzwerk diese temporäre Identifikationsnummer, die er statt seiner eindeutigen IMSI benutzen kann, um sich für die Nutzung von Diensten beim Netzwerk zu identifizieren [TS-03.20].

Um Authentizität von Teilnehmeridentitäten und die Vertraulichkeit von kommunizierten Nutzdaten und Signalisierungsinformationen sicherzustellen, werden in TS-03.20 die Algorithmen A3, A8 und A5 definiert. Eine Beschreibung der Algorithmen findet sich nicht im Standard, die Algorithmen konnten aber durch Reverse Enginnering herausgefunden werden. So veröffentlichten Briceno u. a. [1998] die Implementierung von COMP128, einer Kombination der A3 und A8 Algorithmen und Briceno u. a. [1999] die Implementierung von A5/1 und A5/2. Mit der Entwicklung neuerer Algorithmen hat 3GPP den „Security by Obscurity“ Ansatz fallen gelassen. So steht zum Beispiel die Spezifikation von A5/3 auf der Website von 3GPP zu Verfügung[1]. Die Implementierung von A3 und A8 wurde von 3GPP als netzanbieterspezifisch bezeichnet, COMP128 wäre nur eine Beispielimplementierung gewesen und war nicht für den konkreten Einsatz gedacht. A5/1 und A5/2 wurden von

[1] `http://www.3gpp.org/specifications/60-confidentiality-algorithms`

3GPP auch nachträglich nicht mehr selbst veröffentlicht, finden sich aber online[2]. Die folgenden Abschnitte gehen näher auf die in GSM verwendeten Sicherheitsalgorithmen ein.

3.6.1. GSM Authentifizierung

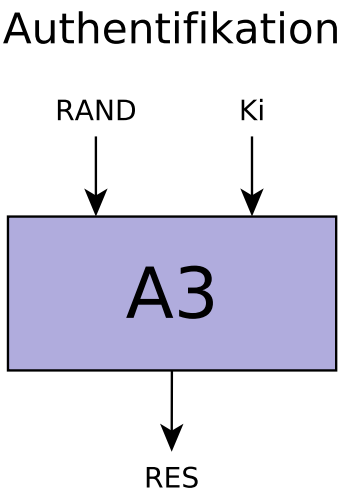


Abbildung 3.15.: Der A3-Algorithmus, erstellt mit yEd nach [TS-03.20, Abb. 3.1]

Die Authentifizierung wird vom Netzwerk gestartet. Zuerst werden vom MSC die für die Überprüfung der Identität des Teilnehmers benötigten Werte mit seiner IMSI beim AuC angefragt. Das AuC generiert die zufällige Challenge RAND und holt sich vom HLR den geheimen Schlüssel Ki für die erhaltene IMSI. Aus den beiden wird mit dem Authentifizierungsalgorithmus A8 die erwartete Antwort SRES des Teilnehmers berechnet [TS-03.20, Kap. 3, Anhang C.2]. Die berechneten Werte werden konkateniert und als Authentifizierungsvektor [RAND ‖ SRES] an das MSC zurückgeschickt. Das MSC kann der MS nun den Authentication-Request mit der vom Netzwerk benutzten Challenge RAND schicken. Erhält die MS einen Authentication-Request, lässt es von der SIM-Karte, die Zugriff auf A8 und Ki hat, das Ergebnis RES berechnen. Die Anfrage vom MSC wird

[2] `http://www.scard.org/gsm/`

dann mit einer Authentication-Response, die RES enthält, beantwortet. Ist der Vergleich RES == SRES im MSC erfolgreich, so ist die MS authentifiziert.

Da das Verfahren keine gegenseitige Authentifizierung zulässt, sondern nur die des Netzteilnehmers, ermöglicht es einem Angreifer eine falschen BTS zu installieren, dessen Identität von der MS nicht überprüft werden kann. Durch die Einführung des UMTS-AKA und der Möglichkeit, dieses auch in GSM-Netzwerken zu verwenden, wurde diese Schwachstelle behoben. Das Verfahren und die mögliche Interoperabilität mit 2G Infrastruktur wird in Unterabschnitt 3.6.3 erklärt.

3.6.2. GSM Schlüsselgenerierung

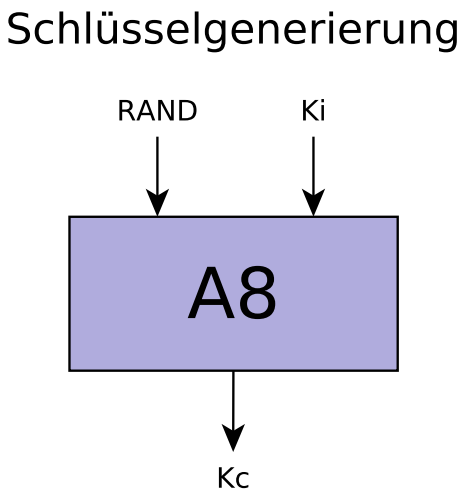


Abbildung 3.16.: Der A8-Algorithmus, erstellt mit yEd nach [TS-03.20, Abb. 4.1]

Nach erfolgreicher Authentifikation des Netzteilnehmers wird, mit dem Algorithmus A8 zur Schlüsselgenerierung, auf beiden Seiten ein 64 Bit langer GSM Cipher Key (Kc) berechnet, der für die weitere Verschlüsselung dieser Verbindung verwendet wird [TS-03.20, Kap. 4.3, Anhang C.3]. Die Eingangsparameter von A8 sind die gleichen wie von A3, vermutlich ein Grund für deren gemeinsame Implementierung in COMP128.

Die Kombination der Verfahren zur Authentifizierung und Schlüsselgenerierung in GSM

wird AKA genannt. Nach der Schlüsselgenerierung wird Kc in MS und BTS als sogenannter Security Context unter einer Cipher Key Sequence Number (CKSN) abgespeichert. Da die CKSN bereits beim CM Service Request mitgeschickt wird, kann die BTS prüfen, ob bereits ein gültiger Security Context mit der MS besteht und diesen wiederverwenden. Das AKA muss also nicht erneut ausgeführt werden, da der kryptografische Schlüssel Kc beiden Seiten bereits bekannt ist.

3.6.3. UMTS Authentication and Key Agreement

Für UMTS wurde mit dem 3G AKA ein gegenseitiges Authentifizierungsverfahren spezifiziert [TS-33.102], um die Schwachstellen der nur einseitigen 2G Authentifizierung zu beheben. Dadurch wird es einem Angreifer erschwert, in einem 3G Netzwerk eine falsche BTS auf der Um-Schnittstelle zu installieren, da die Authentizität des Netzwerks vom Netzteilnehmer überprüft werden kann. Außerdem wurde die Länge des zwischen 3G AuC und UMTS Subscriber Identity Module (USIM) geteilten geheimen Schlüssels K von den in GSM verwendeten 64 Bit auf 128 Bit erhöht. Abbildung 3.17 zeigt die in den Authentication Vector Network (AUTN) einfließenden Werte und Abbildung 3.17 wie die Authentifizierung des Netzwerks in der USIM abläuft. Im UMTS-Standard haben sich einige Bezeichnungen verändert, so wird MS zu Mobile Equipment (ME), Ki zu K, und SRES zu Expected Subscriber Response (XRES).

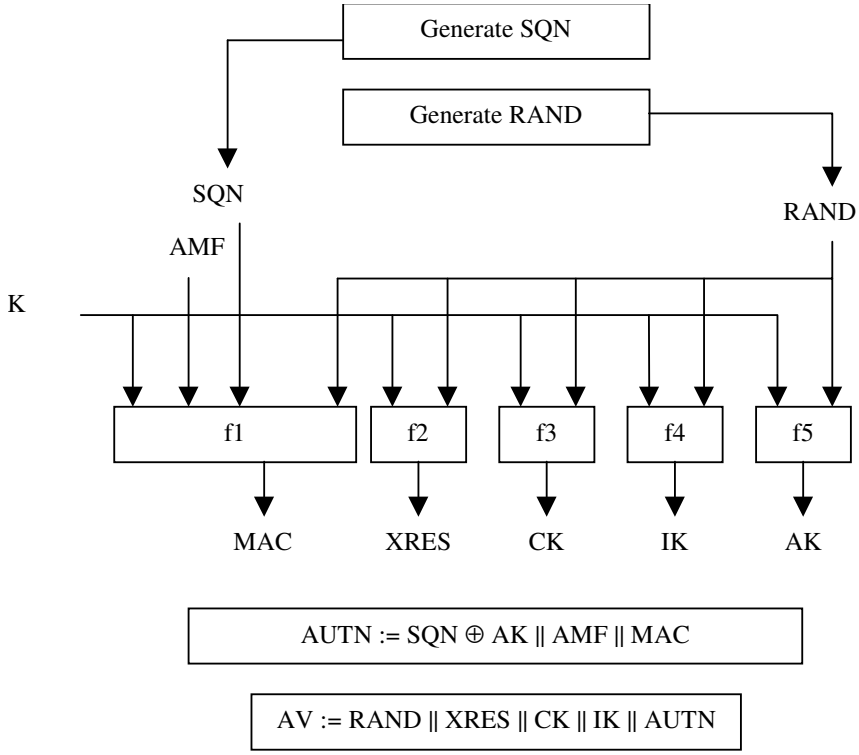


Abbildung 3.17.: Die Generierung des UMTS Authentifizierungsvektors, aus [TS-33.102, Abb. 7]

Wie in GSM erzeugt das AuC zuerst eine zufällige Challenge RAND und eine teilnehmerspezifische Sequence Number (SQN) [TS-33.102, Anhang C.1 und C.2]. Diese Sequenznummer bietet Schutz vor Replay-Attacken mit aufgezeichneten Authentifizierungsabläufen, da das ME nur Anfragen akzeptiert, deren SQN ungefähr gleich der im ME mitgezählten SQN ist. Ein proprietäres Authentication and Key Management Field (AMF) geht ebenfalls in die Berechnung mit ein [TS-33.102, Anhang H]. Das AuC erzeugt dann folgende Werte:

- **Message Authentication Code (MAC)**: $MAC = f1_K(SQN \parallel RAND \parallel AMF)$
- **Expected Subscriber Response (XRES)**: $XRES = f2_K(RAND)$ wobei f1 und f2 Authentifizierungsfunktionen sind.
- **3G Cipher Key (CK)**: $CK = f3_K(RAND)$
- **Integrity Key (IK)**: $IK = f4_K(RAND)$
- **Anonymity Key (AK)**: $AK = f5_K(RAND)$ wobei f3, f4 und f5 Schlüsselerzeu-

gungsfunktionen sind. AK wird benutzt um die Sequenznummer zu anonymisieren und kann auch 0 sein, wenn das nicht erforderlich ist.

Aus diesen wird schließlich der Authentifizierungsvektor AUTN = SQN ⊕ AK ‖ AMF ‖ MAC gebildet und im Authentication-Request an den Mobilfunkteilnehmer ausgeliefert.

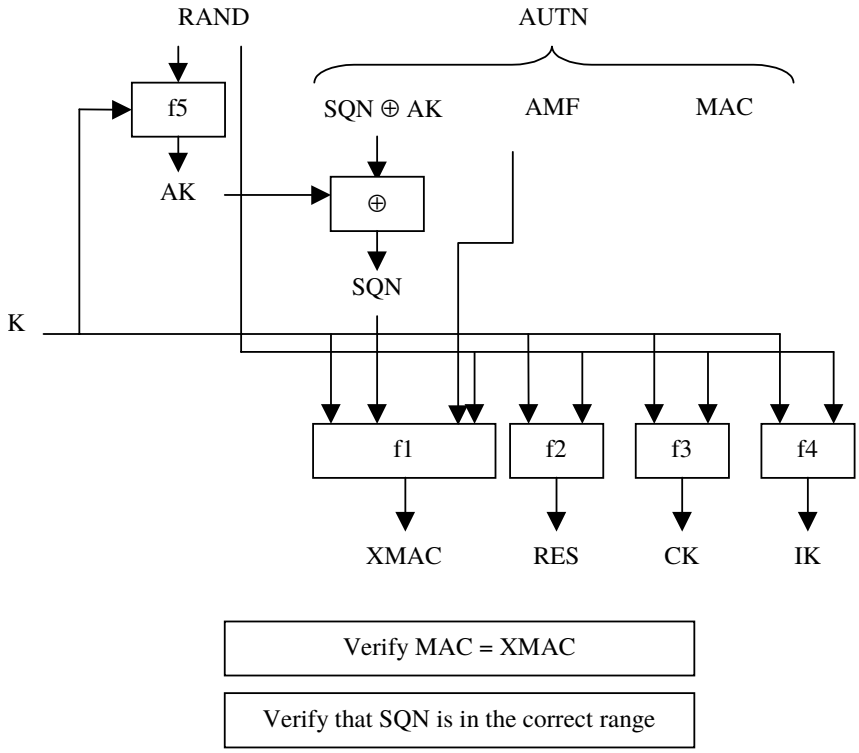


Abbildung 3.18.: Die Authentifizierung des Netzwerks in der USIM, aus [TS-33.102, Abb. 9]

Erhält das ME den Authentication-Request des Netzwerks, so kann es mit der AUTN überprüfen, ob dieses Kenntnis seines geheimen Schlüssels K hat.

Als erstes löst es die Anonymisierung der Sequenznummer auf und holt sich deren Wert mit SQN = (SQN ⊕ AK) ⊕ AK, wobei für die Berechnung von AK die Funktion f5, die auch dem AuC zur Verfügung steht, benutzt wird. Sollte die erhaltene SQN nicht innerhalb der richtigen Größenordnung liegen, wird ein Synchronisationsfehler mit der aktuellen SQN der USIM an das Netzwerk zurückgeschickt und die laufende Authentifizierung abgebrochen.

Mit der Sequenznummer und AMF kann nun aus dem AUTN der erwartete Authentifizierungscode des Netzwerks XMAC berechnet werden. Wenn dieser gleich dem aus

AUTN erhaltenen MAC ist, weiß die USIM, dass das Netzwerk ebenfalls den geheimen Schlüssel K kennt. Das Netzwerk ist somit authentifiziert und die Authentication-Response mit dem Ergebnis RES wird zurückgeschickt. Die anschließende Authentifizierung der Netzteilnehmer läuft wie in GSM durch die Überprüfung RES == XRES des Netzwerks.

Der Schlüssel CK geht dann in den Verschlüsselungsalgorithmus ein und sorgt wie schon in GSM, für Vertraulichkeit der Daten. IK ist in 3G neu und wird verwendet, um mit der Integritätsfunktion f9 [TS-33.102, Kap. 6.5.3] die Integrität der Daten sicherzustellen.

Das für UMTS spezifizierte AKA kann in GSM ebenfalls benutzt werden, sofern der Netzteilnehmer und das AuC die 3G Authentifizierungsverfahren unterstützen. Abbildung 3.19 zeigt die verschiedenen Möglichkeiten der Interoperabilität zwischen 2G und 3G Netzwerk- und ME. Da aktuelle MEs (Mobilfunkgeräte, USIM) und Kernnetzwerke (AuC) alle 3G kompatibel sind, wird das GSM-AKA in der Regel nicht mehr benutzt.

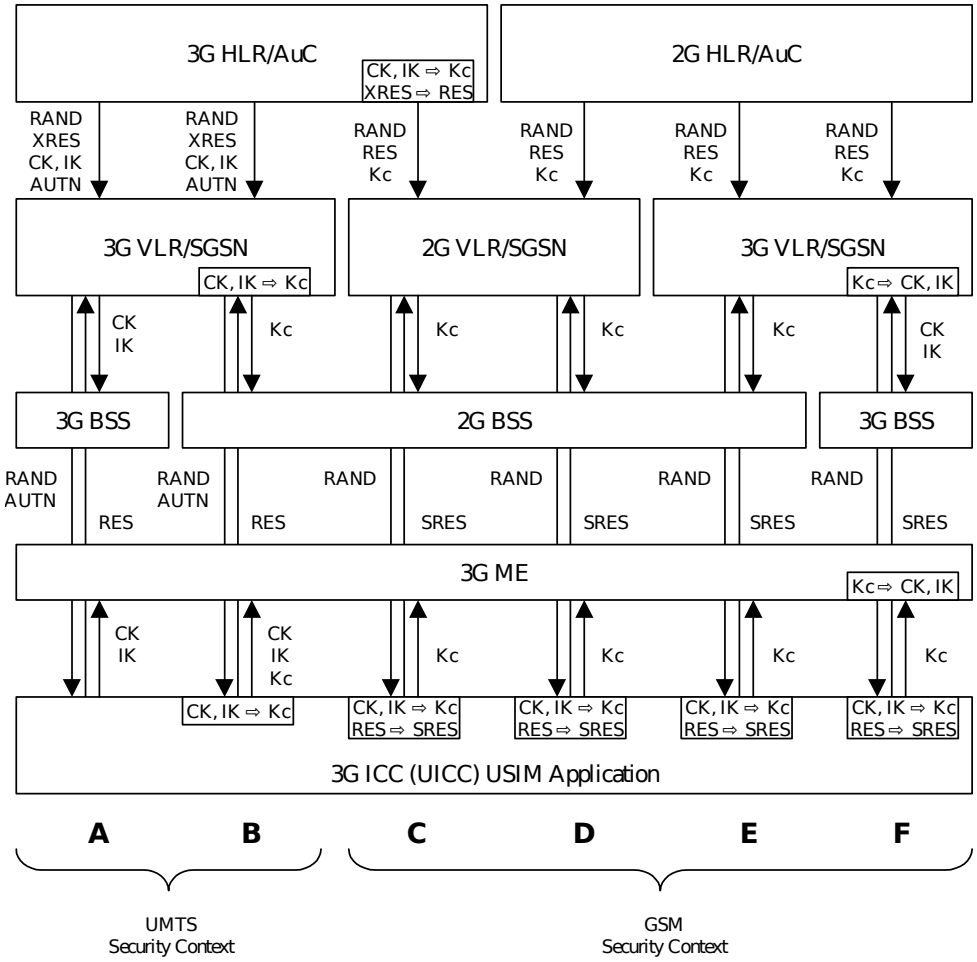


Abbildung 3.19.: Interoperabilität des 3G AKA mit 2G Netzwerken, aus [TR-31.900, Abb. 1]

Die Verwendung des UMTS-AKA in 2G Netzwerken basiert darauf, dass das MSC den vom AuC erhaltenen Wert (RAND oder AUTN) einfach nur in den Authentication-Request einfügt und weiterleitet und SRES nicht selbst berechnet. Von einem 3G-fähigen AuC erhält die USIM also den Wert von AUTN, mit dem sie das Netzwerk authentifizieren kann. Für die Authentication-Response muss die USIM die korrekte Antwort RES berechnen mit der sie, wie im 2G Standard definiert, im MSC authentifiziert wird. Die einzigen Komponenten, die für die Ausführung des UMTS-AKA 3G-fähig sein müssen, sind also USIM und AuC. Der von f3 berechnete kryptografische Schlüssel CK wird von der 2G BTS als Kc verwendet. Für den Integritätsschlüssel IK hat die BTS aber keine Verwendung. Die Integrität der übertragenen Daten bleibt weiterhin ungeschützt.

3.6.4. Verschlüsselung

Bevor die Kommunikation über die Funkschnittstelle verschlüsselt werden kann, muss der verwendete A5-Algorithmus zwischen MS und BTS ausgehandelt werden [TS-03.20, Kap. 4.8], was in der „MM Connection Establishment Prozedur“ [TS-24.008, Kap. 4.5.1.1] genau beschrieben wird. Die A5-Algorithmen sind in der Hardware von Endgerät und BTS realisiert, da sie in Echtzeit verschlüsseln und entschlüsseln müssen.

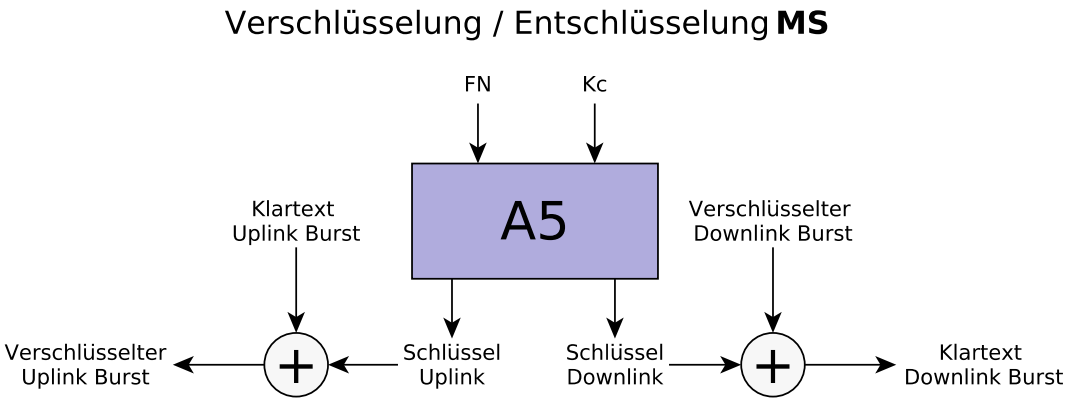


Abbildung 3.20.: Der A5-Algorithmus, erstellt mit yEd nach [TS-03.20, Abb. C.2]

Die MS schickt der BTS im CM Service Request [TS-24.008, Kap. 10.5.1.6] unter anderem das Mobile Station Classmark 2, welches ihre unterstützten Verschlüsselungsverfahren enthält. Die BTS wählt daraus den größten gemeinsamen Nenner aus und informiert die MS im CM Service Accept [TS-24.008, Kap. 9.2.5] über diesen. Durch die ebenfalls im Request übertragene CKSN kann das Netzwerk überprüfen, ob schon ein übereinstimmender Security Context zwischen MS und Netzwerk besteht, oder erst durch die Authentifizierung des Teilnehmers erstellt werden muss. Ist der beste gemeinsam verfügbare Algorithmus gefunden und Kc bekannt, kann die Verbindung mit diesem geschützt werden. Die FN, die mit in die Schlüsselstromgenerierung einfließt, verhindert Replay-Angriffe. Da diese nach einem Hyperframe allerdings wieder bei 0 beginnt und zumindest in einem laufenden Gespräch kein neuer Security Context zwischen MS und BTS ausgehandelt wird, wiederholt sich etwa alle drei Stunden und 26 Minuten der verwendete Schlüsselstrom. Für ein Gespräch, welches länger dauert entsteht dadurch ein Sicherheitsproblem, da der

Schlüsselstrom wiederverwendet wird.

$$(A \oplus C) \oplus (B \oplus C) = (A \oplus B) \oplus (C \oplus C) = A \oplus B \tag{3.9}$$

Gleichung 3.9 zeigt, dass bei Verwendung des gleichen Schlüsselstroms C für zwei verschiedene Klartextdatenströme A und B, deren XOR-Kombination herausgefunden werden.

$$\begin{aligned} &\text{A bekannt} \Rightarrow A \oplus A \oplus B = B \\ &\text{B bekannt} \Rightarrow B \oplus B \oplus A = A \end{aligned} \tag{3.10}$$

Es gibt verschiedene Ansätze und Möglichkeiten, die Datenströme wieder voneinander zu trennen [Borisov u. a., 2001, Kap. 3]. Gleichung 3.10 zeigt die Idee dahinter. Aus bekannten Teilen von A kann direkt der Inhalt dieses Teils von B hergeleitet werden und umgekehrt.

Alle A5 Verschlüsselungsverfahren sind Stromchiffren und basieren auf der binären Addition des Datenstroms mit einem generierten Schlüsselstrom. In Anhang, in Abschnitt B.3 werden die von 3GPP für GSM spezifizierten A5-Verschlüsselungsverfahren kurz aufgeführt.

Laut Untersuchungen von gsmmap.org [2016] werden in deutschen 2G Netzen aktuell nur die Algorithmen A5/1 und A5/3 unterstützt, von denen beide bekannte Schwachstellen haben. Der aktuell (Stand 2017) als sicher geltende Algorithmus A5/4 [TS-55.226, 2011] ist zwar seit einigen Jahren spezifiziert, wird aber nicht verwendet.

	O2	**Eplus**	**Vodafone**	**Telekom**
A5/1	73%	56%	41%	26%
A5/3	27%	44%	59%	74%

Tabelle 3.4.: Die prozentuale Verteilung der A5-Algorithmen in Deutschland, aus [gsmmap.org, 2016]

3.6.5. Zusammenspiel der Sicherheitsalgorithmen in GSM

Abbildung 3.21 stellt das Zusammenspiel der Sicherheitsalgorithmen in GSM dar. Aus Ki und der im AKA erhaltenen Challenge RAND, berechnet das MS mit A3 das Ergebnis RES und schickt es an das Netzwerk zurück. Das Netzwerk vergleicht das Ergebnis des MS mit dem eigenen als Authentizitätsprüfung. Aus den gleichen Eingangsparametern berechnet A8 auf beiden Seiten den Schlüssel Kc. Nach erfolgreicher Authentifizierung und

Aktivierung der Verschlüsselung wird aus Kc und der aktuellen FN der Schlüsselstrom für Uplink und Downlink generiert. Durch die XOR-Verknüpfung mit den 114 verschlüsselten Nutzdatenbits eines dekodierten, eingehenden Burst kann dieser entschlüsselt werden. Durch die Verknüpfung mit dem unverschlüsselten Nutzdatenbits eines ausgehenden Bursts wird dieser verschlüsselt.

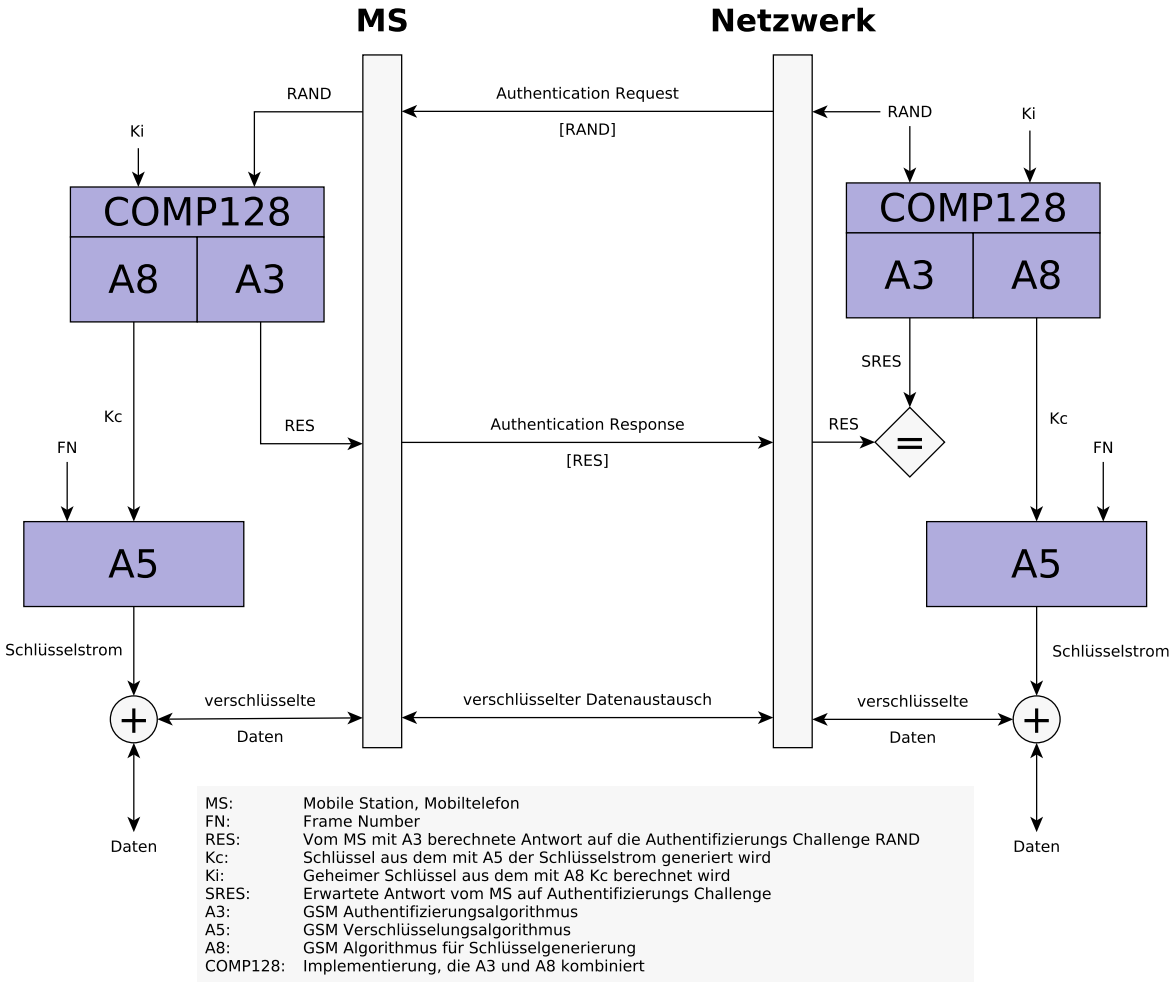


Abbildung 3.21.: Das Zusammenspiel der GSM Sicherheitsalgorithmen, erstellt mit yEd

3.7. Timing Advance

Da die Datenübertragung, abhängig von der Entfernung zwischen MS und BTS, unterschiedlich lange dauert, muss regelmäßig die Signallaufzeit gemessen und daraus der Timing Advance (TA)-Wert [TS-04.04, Kap. 6.1] berechnet werden. Dieser bestimmt, wie viel früher die MS ihre Daten losschicken muss, damit sie zum richtigen Zeitpunkt bei der BTS ankommen. Wird kein Timing-Advance verwendet, stimmt der Zeitpunkt der Ankunft des Signals unter Umständen nicht mit dem Anfang eines TDMA-Zeitschlitzes überein. Damit liegt die Nachricht außerhalb eines gültigen physikalischen und logischen Kanals und kann von der BTS nicht richtig dekodiert werden. Auf der Seite des MS wird die zeitliche Verzögerung bei der Synchronisation mit der BTS berücksichtigt, weshalb dort die Nachrichten in den korrekten Zeitschlitzen ankommen.

Der erste TA-Wert wird dem MS in der „Immediate Assignment“ Nachricht auf dem AGCH mitgeteilt und während einer Verbindung auf dem SACCH aktualisiert. Der TA-Wert ist quantisiert durch Vielfache der Bitübertragungsdauer von $3,7\mu s$. Er kann Werte zwischen 0 und 63 annehmen, wodurch sich der Sendezeitpunkt maximal um $63 \cdot 3.7\mu = 233\mu s$ verschieben lässt. Das entspricht etwas weniger als einem halben Burst. Das elektromagnetische Funksignal breitet sich mit Lichtgeschwindigkeit aus. In der Zeit von $233\mu s$ wird also ein Weg von ca. $300.000^{km}/_{s} \cdot 233 \cdot 10^{-6} s = 69.9km$ zurückgelegt, womit sich ein maximaler Abstand vom MS zur BTS von etwa $35km$ realisieren lässt.

4. Osmocom

Das **O**pen **S**ource **MO**bile **COM**munications Projekt ist ein Umbrella-Projekt für freie und quelloffene Software im Bereich der mobilen Telekommunikation. Es umfasst Software und Werkzeuge, die eine Vielzahl von mobilen Kommunikationsstandards wie GSM, UMTS, Digital Enhanced Cordless Telecommunications (DECT) und Terrestrial Trunked Radio (TETRA) implementieren [osmocom.org, 2017c].

Die unter Osmocom vereinigten Projekte werden vom Unternehmen sysmocom[1] und freien Entwicklern aktiv gepflegt. Der Gründer von Osmocom, wie auch sysmocom ist Harald Welte. Es gibt eine aktive Community, Mailinglisten und in Redmine gepflegte Repositories der verschiedenen Projekte.

Die Software findet im akademischen Bereich Verwendung. Aufgrund des quelloffenen Codes und der vielen Anwendungsmöglichkeiten eignet sie sich, um einen Einblick in die Welt der mobilen Kommunikation zu bekommen [osmocom.org, 2017e].

Kommerziell genutzt werden die Projekte osmoBSC und osmoBTS zum Beispiel von Rhizomatica[2], um in Mexiko Bergregionen an ein GSM-Netz anzubinden [osmocom.org, 2017b]. In der maritimen Telekommunikation bietet On-Waves[3] auf OsmoBTS, OsmoPCU und OsmoBSC basierende Lösungen an, um Schiffe mit einem GSM-Netz abzudecken [Nomine, 2017]. Für zwei aktuelle Projekte von Facebook wird ebenfalls Osmocom Technologie verwendet. OpenCellular nutzt osmoBTS und hat die Entwicklung einer robusten Basisstation für den Einsatz in klimatisch anspruchsvollen Regionen zum Ziel [McMilin und Ali, 2017]. Der „Community Cellular Manager“ soll ein Verwaltungssystem und Abrechnungsmöglichkeiten für mehrere privat verwaltete Netzwerkzellen liefern und implementiert eine Schnittstelle zu osmoBSC [Ramadan, 2017].

[1] `https://www.sysmocom.de/`
[2] `https://www.rhizomatica.org`
[3] `https://www.on-waves.com`

Es gibt mehrere Osmocom Bibliotheken, die in Projekten gemeinsam genutzte Funktionen beinhalten [osmocom.org, 2017d]. Für Debian sind aktuelle Nightly Builds der am häufigsten benutzten Bibliotheken verfügbar [osmocom.org, 2017a].

Die Projekte, die für den Aufbau eines funktionierenden GSM-Netzwerks benötigt werden, werden in den folgenden Kapiteln erklärt.

4.1. OsmocomBB

OsmocomBB ist eine Open-Source GSM-Baseband Implementierung. Das Projekt wurde mit dem Ziel gestartet, proprietäre GSM-Baseband Software vollständig zu ersetzen. Es implementiert Treiber für interne und externe, analoge und digitale GSM Baseband Peripheriegeräte und den MS seitigen GSM-Protokollstapel von Schicht 1 bis 3. [osmocom.org, 2017f]

Mit OsmocomBB ist es also möglich, auf der Basis von freier Software Mobilfunkdienste (Anrufe, SMS) auf einem kompatiblem Handy zu nutzen.

Alle Informationen zum Projekt findet man im Osmocom Wiki[4].

4.2. OsmoBTS

OsmoBTS realisiert eine GSM-BTS in Open-Source. Damit implementiert das Projekt den Protokollstapel der Abis-Schnittstelle zum BSC und den netzwerkseitigen Protokollstapel der Um-Schnittstelle. Was das Projekt besonders macht ist die Unterstützung einer Vielzahl verschiedener Transceiverhardware und Implementierungen der physikalischen Schicht. OsmoBTS kann zum Beispiel zusammen mit OsmoTRX verwendet werden, um Software Defined Radio (SDR)-Geräte, wie die Universial Software Radio Peripherals (USRPs) aus dem Angebot von Ettus Research[5], als Transceiver zu verwenden. [ftp.osmocom.org, 2017, OsmoBTS Benutzerhandbuch]

[4] `https://osmocom.org/projects/baseband/wiki`
[5] `https://www.ettus.com/`

Die Konfiguration des BTS ist zur Laufzeit über ein Virtual Teletype (VTY) Command Line Interface (CLI) möglich.

Alle Informationen zum Projekt findet man im Wiki[6] und im Benutzerhandbuch.

4.3. OsmoBSC und OsmoNITB

OsmoBSC implementiert einen GSM-BSC und somit den Protokollstapel der Abis-Schnittstelle zur BTS und der A-Schnittstelle zum MSC. [ftp.osmocom.org, 2017, OsmoBSC Benutzerhandbuch]

Alle Informationen zu OsmoBSC findet man im Wiki[7] und im Benutzerhandbuch.

OsmoNITB implementiert einen GSM-BSC als „Network in the Box“. Das Projekt realisiert die vollständige Abis-Schnittstelle zur BTS, enthält jedoch intern alle nötigen Netzkomponenten und implementiert daher keine A-Schnittstelle. Mit dem NITB ist es möglich ein in sich funktionales Netzwerk, bestehend aus dem NITB selbst und einer Reihe von angebundenen BTS, aufzubauen. [ftp.osmocom.org, 2017, OsmoNITB Benutzerhandbuch]

Alle Informationen zu OsmoNITB findet man im Wiki[8] und im Benutzerhandbuch.

Die Konfiguration von BSC und NITB ist zur Laufzeit über ein VTY CLI möglich.

[6] https://osmocom.org/projects/osmobts/wiki/Wiki
[7] https://osmocom.org/projects/openbsc/wiki/Osmo-bsc
[8] https://osmocom.org/projects/osmonitb/wiki/OsmoNITB

5. Die theoretische Ausarbeitung des Angriffs

In diesem Kapitel wird der MitM Angriff theoretisch ausgearbeitet und gezeigt, unter welchen Voraussetzungen er in GSM umgesetzt werden kann. Die Schwachstellen im GSM-Standard, die den Angriff möglich machen werden dargestellt und die Schritte herausgearbeitet, die notwendig sind um ein ausgehendes Telefonat erfolgreich umzuleiten.

5.1. Das Problem von Stromverschlüsselung ohne Integrität

GSM verwendet Stromverschlüsselung zum Schutz der vertraulichen Kommunikation und ist damit anfällig für „Known Plaintext“ Angriffe ohne Kenntnis des kryptografischen Schlüssels. Da die Integrität der übertragenen Nachrichten nicht geschützt, ist kann ein MitM-Angreifer bekannte Daten beliebig verändern.

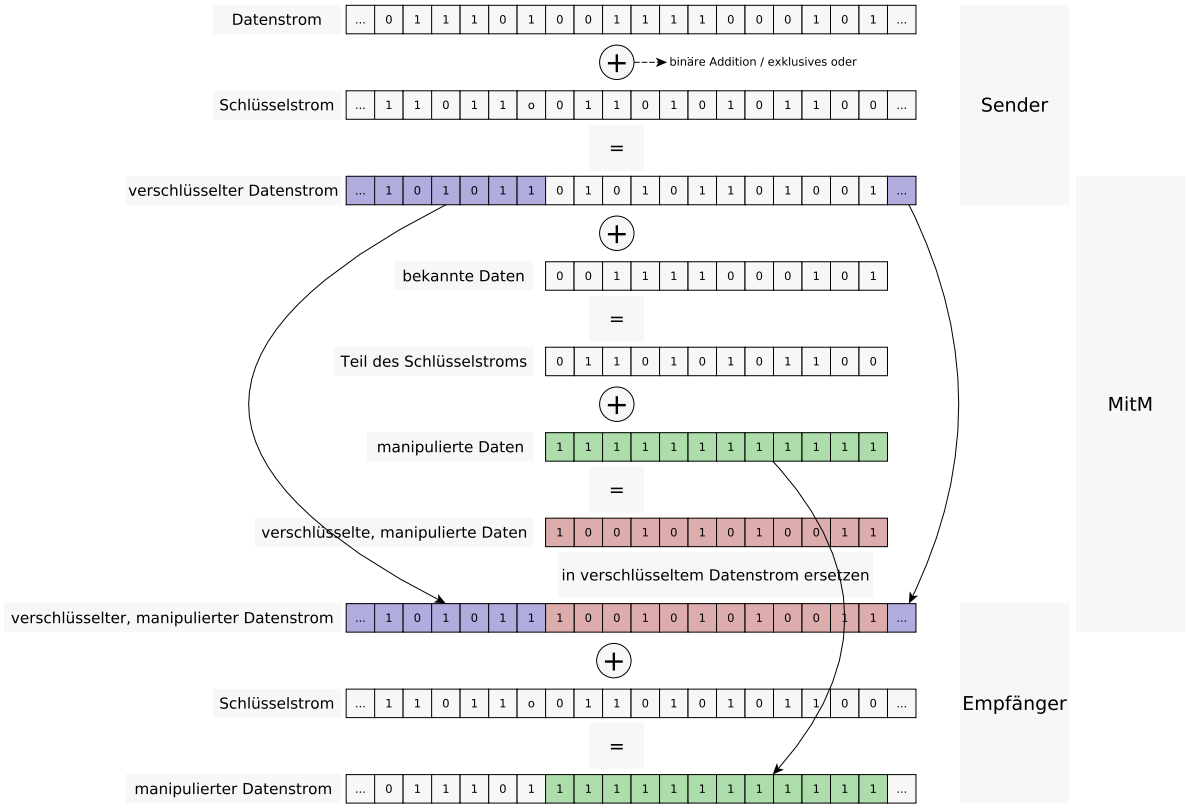


Abbildung 5.1.: Datenmanipulation ohne Kenntnis des kryptografischen Schlüssels, erstellt mit yEd

Abbildung 5.1 zeigt an einem Ausschnitt des Datenstroms, wie bekannte Daten trotz Verschlüsselung von einem MitM geändert werden können. Da der Chiffrestrom das Ergebnis der XOR-Kombination von Datenstrom und Schlüsselstrom ist, kann der Angreifer für bekannte Daten, den verwendeten Schlüsselstrom berechnen. Die Kenntnis des Schlüsselstroms für einen Teil des Datenstroms erlaubt dem Angreifer, diesen Teil beliebig zu manipulieren. Beim Empfänger kommt nach der Entschlüsselung der geänderte Datenstrom heraus. Methoden zum Schutz der Integrität übertragen zusätzlich zu den Nutzdaten in der Regel einen Hashwert, der nur korrekt berechnet werden kann, wenn der vollständige Klartext bekannt ist. Änderungen eines Angreifers, der nur einen Teil der Daten kennt, können also vom Empfänger bemerkt werden. Ohne Methoden zum Schutz der Integrität kann der Empfänger die Manipulation hingegen nicht bemerken.

Die Kenntnis über den mangelnden Schutz von Verschlüsselung ohne Integrität ist weit verbreitet. Dennoch werden darauf basierende Angriffe von verschiedenen Übertragungsprotokollen und deren Spezifikationen ermöglicht. So zeigen Paterson und Yau [2006]

die Schwachstelle in der Internet Protocol Security (IPSec) Implementierung im Linux Kernel auf. Sie können, wenn IPSec ohne Authentifizierung und Integrität konfiguriert ist, die Verschlüsselung von Ecapsulated Security Payload (ESP) Paketen erfolgreich brechen. Degabriele und Paterson [2007] stellen mehrere implementierungsunabhängige Angriffe auf IPSec vor und zeigen damit, dass die Schwachstelle in IPSec nicht nur im Linux Kernel existiert, sondern in der IPSec-Standardisierung. Der für den Standard IEEE 802.11 für Drahtlosnetzwerke spezifizierte Verschlüsselungsalgorithmus Wired Equivalent Privacy (WEP), leidet an der gleichen Schwachstelle. Er gilt heute offiziell als unsicher, von Herstellern wird generell empfohlen, WEP nicht mehr für ihre Geräte zu verwenden. Dennoch besteht nach wie vor die Möglichkeit, die meisten Wlan-Router und andere IEEE 802.11 kompatible Geräte für WEP zu konfigurieren. Bittau u. a. [2006] stellen einen Fragmentierungsangriff auf WEP vor, der wegen seiner kurzen Ausführungszeit, trotz regelmäßiger Erneuerung des Schlüssels funktioniert. Wiederum basiert er auf fehlender Integrität von durch Stromverschlüsselung geschützten Daten. Die Arbeiten weisen alle auf die Gefahren der Nutzung von Verschlüsselung ohne Integrität hin und empfehlen, solche Konfigurationen generell nicht zu unterstützen und zu verwenden.

5.2. Die Idee des Man-in-the-Middle Angriffs auf GSM

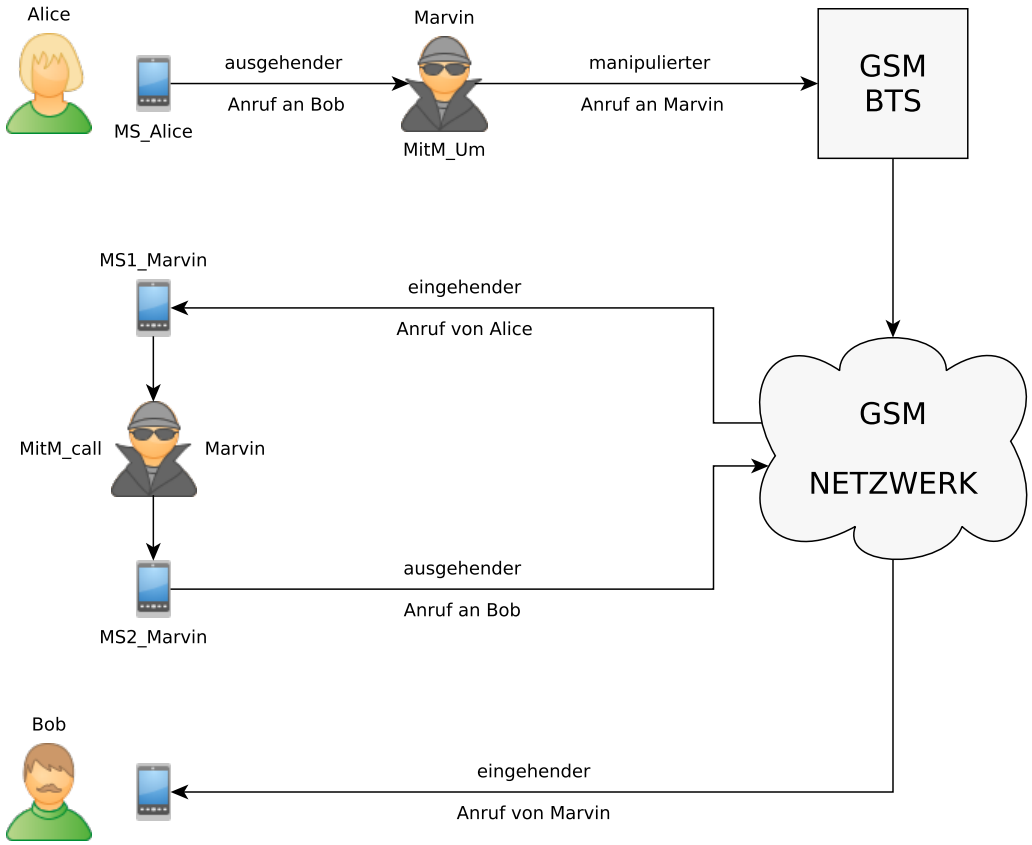


Abbildung 5.2.: Die Idee des MitM-Angriffs, erstellt mit yEd

Das Szenario (siehe Abbildung 5.2):

Alice ist mit dem GSM-Netz verbunden und möchte Bob anrufen. Der Angreifer Marvin hat ein MitM-Gerät (MitM_{Um}) auf der Um Schnittstelle installiert, mit dem er auf physikalischer Ebene Zugriff auf alle ausgetauschten Nachrichten zwischen Alice's Telefon und der GSM-BTS hat. Die Nachrichten sind kodiert und stromverschlüsselt. Stromverschlüsselung schützt nicht vor Datenmanipulation bei bekanntem Klartext und Bob's Telefonnummer ist Marvin bekannt. Er kann deshalb den Anrufaufbau im MitM_{Um} manipulieren und Bob's Telefonnummer austauschen. Da die Integrität der ausgetauschten Nachrichten in GSM nicht geschützt ist, können weder Alice noch die BTS die Manipulation bemerken. Der manipulierte Anruf wird deshalb vom Netzwerk als gültig erkannt und an Marvin's

Mobiltelefon weitergeleitet. Das Netzwerk übernimmt Kodierung und Verschlüsselung des übertragenen Datenstroms, Marvin erhält das Gespräch von Alice im Klartext. Marvin ruft nun Bob an und verknüpft die beiden Anrufe so, dass Bob die Gesprächsdaten von Alice erhält und umgekehrt. Da Marvin Zugriff auf den Klartext beider Datenströme hat, kann er diese aufnehmen und manipulieren. Der dadurch entstandene MitM-Angriff (MitM_{call}) funktioniert unabhängig vom verwendeten Verschlüsselungsverfahren und ist in Abbildung 5.3 dargestellt.

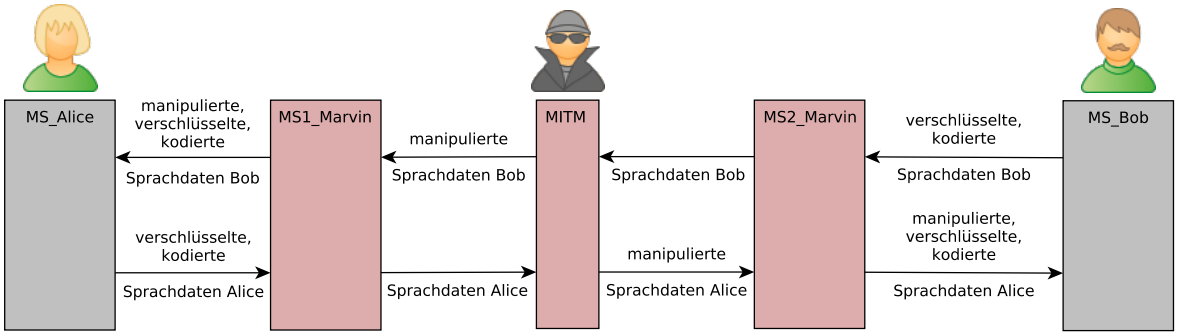


Abbildung 5.3.: Der entwickelte MitM-Angriff auf eine Sprachverbindung, erstellt mit yEd

Dass ein MitM-Angriff auf dem Um möglich ist, haben bereits Barkan u. a. [2003], Meyer und Wetzel [2004] und Paget [2010] gezeigt. Für ihre MitM-Angriffe verwenden sie eine falsche Basisstation (siehe Kapitel 8). Das MitM-Gerät für den vorgestellten Angriff benötigt nur Zugriff auf den verschlüsselten, kodierten Datenverkehr und muss den kryptografischen Schlüssels nicht berechnen. Statt einer falschen Basisstation reicht also ein MitM-Gerät auf physikalischer Ebene, mit geringer Rechenleistung aus. Das würde in etwa der Funktionalität eines GSM-Repeaters mit zusätzlichen Verarbeitungsroutinen für den Datenstrom entsprechen. Statt Daten nur zu verstärken und weiterzuleiten kann er diese auch analysieren und manipulieren. Die Zeit dafür kann durch die Verzögerung des Datenverkehrs von bis zu 233μs gewonnen werden, die die Timing Advance Funktion des GSM-Standards erlaubt (siehe Abschnitt 3.7). Das MitM-Gerät in der Funkschnittstelle, mit Zugriff auf den verschlüsselten und kodierten Datenverkehr, wird in dieser Arbeit vorausgesetzt. Die Möglichkeiten der Nutzung eines modifizierten GSM-Repeaters für den MitM-Angriff werden nicht untersucht.

5.3. Analyse des Nachrichtenflusses auf der Um Schnittstelle beim Anrufaufbau

Um herauszufinden, welche Nachrichten für den Aufbau einer Telefonverbindung zuständig sind, wird der Nachrichtenfluss auf der Um Schnittstelle analysiert. Folgende Fragen müssen beantwortet werden:

- Wie wird das Opfer identifiziert?
- Wie wird die für den Anrufaufbau zuständige „Setup“ Nachricht im Nachrichtenfluss identifiziert?

Abbildung 5.4 und Folgende enthalten die zwischen MS und BTS ausgetauschten Signalisierungsnachrichten. Aktuelle (Stand 2015) Informationen zum untersuchten Nachrichtenfluss finden sich in TS-23.108, Kap. 7.3.2 und TS-24.007, Abb. A.1. Übertragene RR-Signalisierungsnachrichten sind in TS-04.18, MM und CM-Nachrichten, sowie Service-Requests in TS-24.008 spezifiziert.

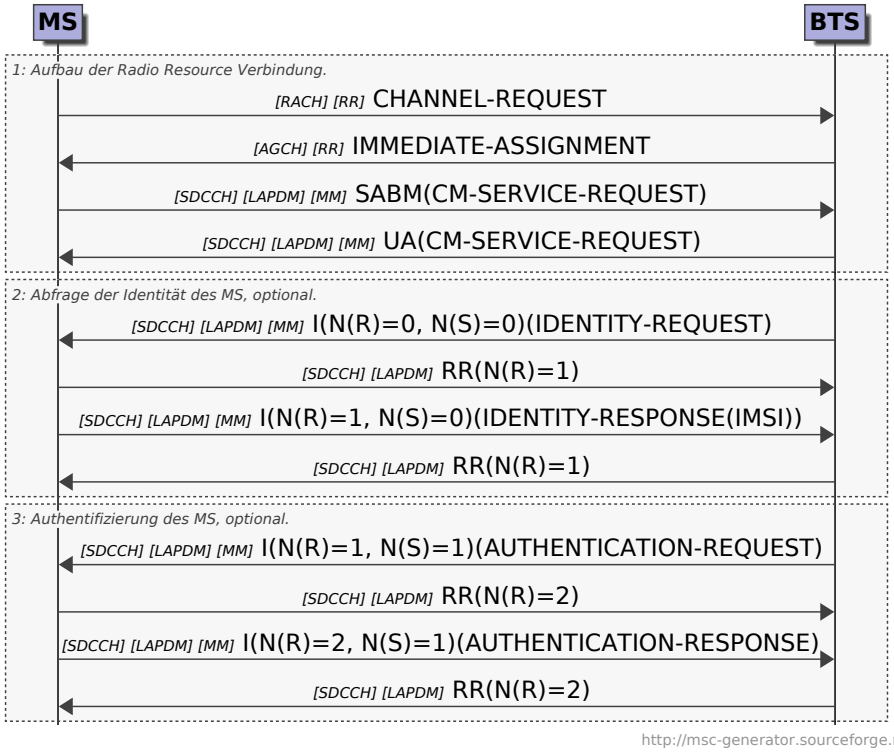


Abbildung 5.4.: Nachrichtenfluss auf der Um-Schnittstelle beim Anrufaufbau, Teil 1, nach [TS-23.108, Kap. 7.3.2] und [TS-24.007, Abb. A.1]

1: Das MS beantragt mit dem `CHANNEL-REQUEST` [TS-04.18, Kap. 9.1.8] einen dedizierten Kanal. Der Request enthält ein Flag, das dem Netzwerk signalisiert, für was der Kanal verwendet werden soll. Bei „Very Early Assignment“ reserviert das Netzwerk an dieser Stelle einen TCH. Das bedeutet, die folgende Signalisierung würde auf dem FACCH und nicht wie in der Abbildung auf dem SDCCH erfolgen [TS-23.108, Kap. 7.3.2]. Mit dem `IMMEDIATE-ASSIGNMENT` [TS-04.18, Kap. 9.1.18] erhält das MS vom Netzwerk die benötigten Informationen zum reservierten Kanal, was es ihm ermöglicht, auf diesen zu wechseln. Nach dem Wechsel in den „Dedicated Mode“ beantragt das MS auf dem SDCCH mit einem SABM-Frame den Wechsel vom unacknowledged in den asynchronous balanced LAPDm-Modus. Damit initiiert es den Aufbau einer von der Datensicherungsschicht geschützten Verbindung (siehe Unterabschnitt 3.3.3). Das SABM-Frame trägt den `CM-SERVICE-REQUEST` [TS-24.008, Kap. 9.2.9] mit der IMSI oder TMSI des Netzteilnehmers, dem Typ des angeforderten CM-Dienstes und Informationen zu MS-seitig unterstützten Verschlüsselungsverfahren mit sich. Der Empfang der SABM-Nachricht und der netzwerkseitige Wechsel in den asynchronous balanced Mode, wird durch ein UA-Frame mit einer Kopie

des erhaltenen Service-Requests bestätigt. Alle weiteren Signalisierungsnachrichten auf dem dedizierten Kanal sind in I-Frames verpackt und werden bei erfolgreichem Empfang mit einem RR-Frame bestätigt. Das RR-Frame hat in dem Fall nichts mit Radio Resource zu tun, sondern ist eine Kontrollnachricht des LAPDm-Protokolls. Die Werte der Zähler N(R) und N(S) für empfangene und gesendetet Nachrichten sind im Flussdiagramm für I und RR-Frames dargestellt.

2: Die Identitätsabfrage des Netzteilnehmers [TS-24.008, 2005, Kap. 4.3.3] erfolgt, wenn das Netzwerk die TMSI aus dem `CM-SERVICE-REQUEST` keiner IMSI zuweisen kann. Sie sollte nur wenn nötig ausgeführt werden, um die Anonymität der Netzteilnehmer zu schützen, also in der Regel nicht bei jeder Anfrage eines Teilnehmers. Der `IDENTITY-REQUEST` [TS-24.008, 2005, Kap. 9.2.10] enthält den Identitätstyp, der abgefragt wird (IMSI, TMSI, IMEI). Das Netzwerk kann mehr als einen Identity Request schicken um weitere Identitäten anzufordern. Nachdem die MS den Request mit einer `IDENTITY-RESPONSE` [TS-24.008, 2005, Kap. 9.2.11] beantwortet und die gewünschte Identität geliefert hat, ist die Routine abgeschlossen.

3: Wie die Identitätsabfrage erfolgt auch die Authentifizierung nur bei Bedarf. Aus dem `CM-SERVICE-REQUEST` kennt das Netzwerk die aktuellen Sicherheitsparameter des MS und kann mit diesen überprüfen, ob ein gültiger Security-Context besteht (siehe Unterabschnitt 3.6.2). Existiert kein gültiger Security-Context muss das AKA erneut durchgeführt werden, ansonsten kann die Authentifizierung übersprungen werden. Wenn MS und AuC UMTS-fähig sind, wird das UMTS-AKA ausgeführt (siehe Unterabschnitt 3.6.3), andernfalls das GSM-AKA (siehe Unterabschnitt 3.6.1). Die ausgetauschten Nachrichten unterscheiden sich dabei nur in der Interpretation des Inhalts von RES und RAND. Das Netzwerk schickt dem MS im `AUTHENTICATION-REQUEST` [TS-24.008, 2005, Kap. 9.2.2], je nach verwendetem AKA, die GSM oder UMTS Challenge RAND. Das MS berechnet die Antwort auf die Challenge und schickt sie dem Netzwerk in der `AUTHENTICATION-RESPONSE` [TS-24.008, 2005, Kap. 9.2.2] zurück. Ist die Authentifizierung erfolgreich, kennen beide Seiten auch den synchronen, kryptografischen Schlüssel für das A5 Verschlüsselungsverfahren.

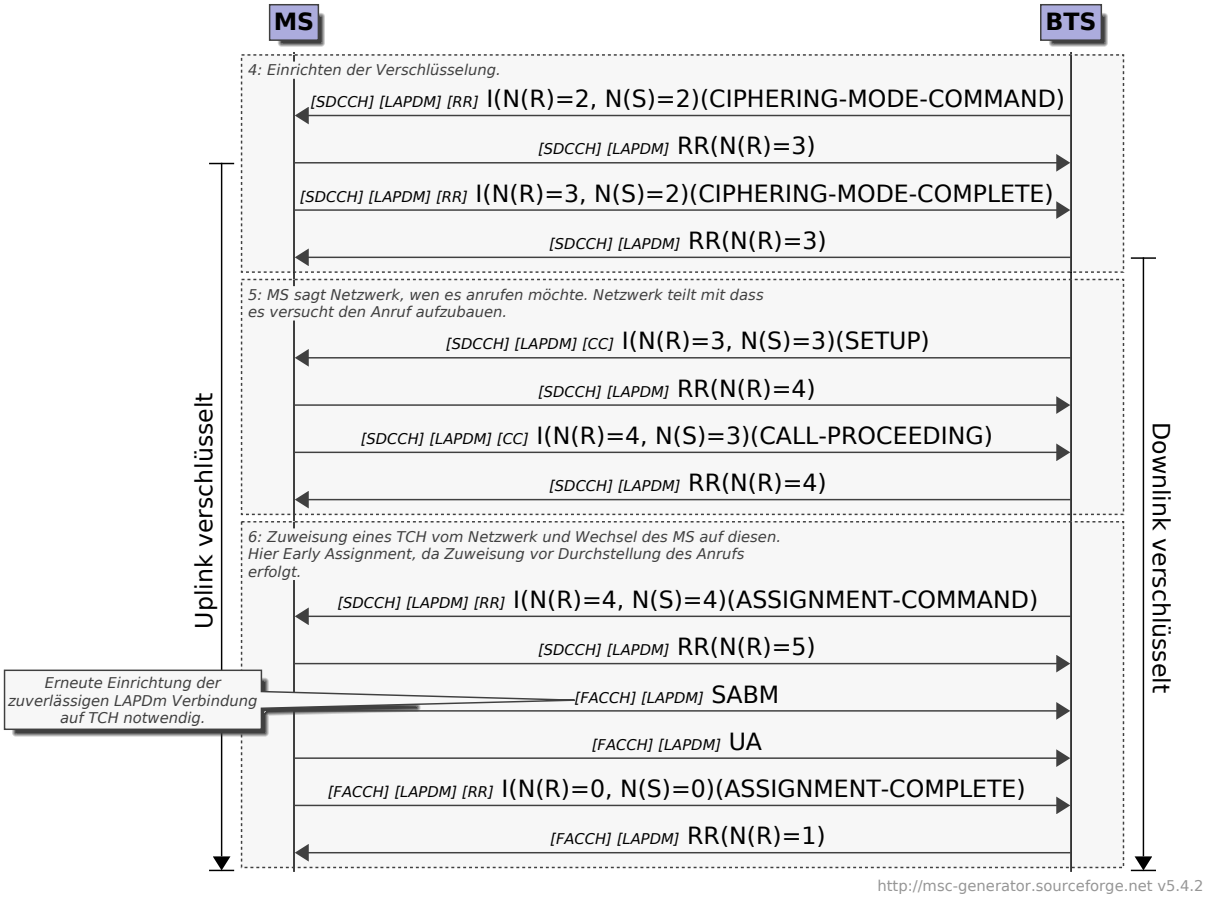


Abbildung 5.5.: Nachrichtenfluss auf der Um-Schnittstelle beim Anrufaufbau, Teil 2, nach [TS-23.108, Kap. 7.3.2] und [TS-24.007, Abb. A.1]

4: Mit dem `CIPHERING-MODE-COMMAND` [TS-04.18, 2006, Kap. 9.1.9] befiehlt das Netzwerk dem MS, von jetzt an die Verschlüsselung des Uplinks zu starten. Sobald die korrekt verschlüsselte `CIPHERING-MODE-COMPLETE` [TS-04.18, 2006, Kap. 9.1.10] Bestätigung des MS empfangen wird, beginnt das Netzwerk den Downlink zu verschlüsseln. Der darauf folgende Nachrichtenverkehr auf dem dedizierten Kanal ist bidirektional durch das ausgehandelte, synchrone Verschlüsselungsverfahren (zum Beispiel A5/3) geschützt.

5: Mit dem Call `SETUP` [TS-24.008, 2005, Kap. 9.3.23.2] wird der Anruf vom MS initiiert. Die angerufene Nummer und weitere Informationen zum Anruf werden dem Netzwerk erst in dieser Nachricht mitgeteilt. Um den Anruf umzuleiten, muss also die `SETUP`-Nachricht manipuliert werden. Mit `CALL-PROCEEDING` [TS-04.18, 2006, Kap. 9.3.3] wird dem MS signalisiert, dass das Netzwerk das `SETUP` erhalten hat und den

Anruf bearbeitet.

6: Bei „Early Assignment“ erfolgt, direkt nach dem `CALL-PROCEEDING`, der Wechsel vom SDCCH auf den TCH. Er wird durch einen `ASSIGNMENT-COMMAND` [TS-04.18, 2006, Kap. 9.1.2] initiiert. Auf dem TCH wird für die weitere Signalisierung eine zuverlässige Verbindung benötigt, die mit den LAPDm-Frames SABM und UA aufgebaut wird. Die Frames tragen diesmal keine Informationen höherer Schichten mit sich, der `CM-SERVICE-REQUEST` wurde ja bereits übertragen. Mit dem `ASSIGNMENT-COMPLETE` teilt das MS dem Netzwerk mit, dass es erfolgreich den Kanal gewechselt hat.

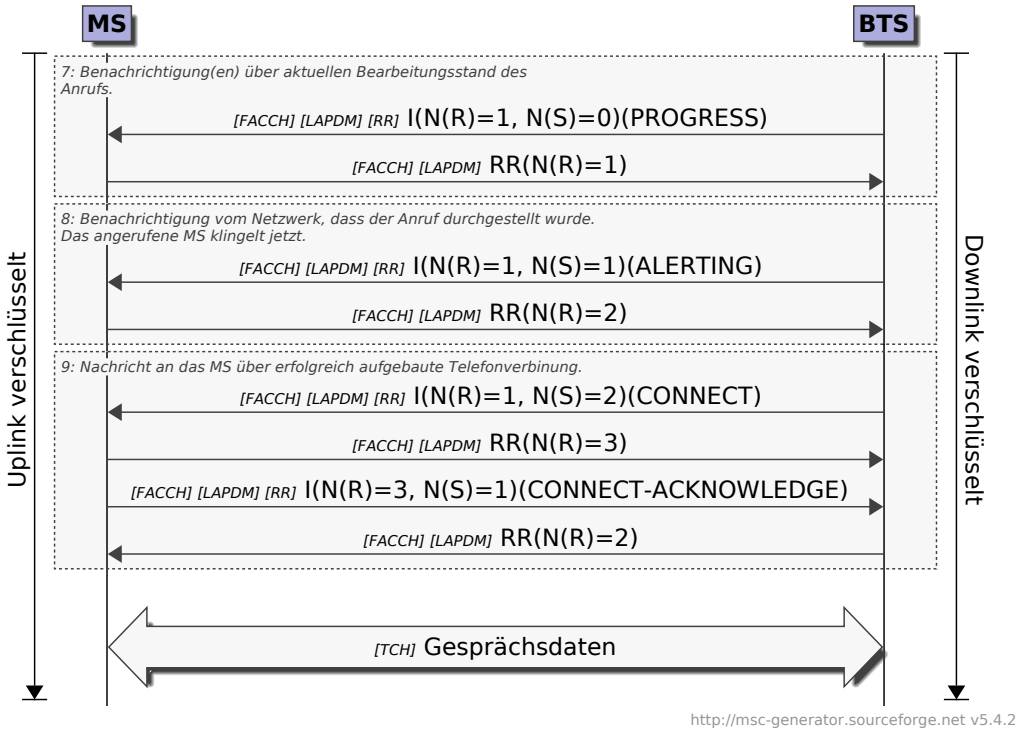


Abbildung 5.6.: Nachrichtenfluss auf der Um-Schnittstelle beim Anrufaufbau, Teil 3, nach [TS-23.108, Kap. 7.3.2] und [TS-24.007, Abb. A.1]

7: Mit `PROGRESS`-Nachrichten [TS-24.008, 2005, Kap. 9.3.17] wird das MS mit Informationen zum aktuellen Stand der Bearbeitung des Anrufs versorgt. `PROGRESS` kann während des Anrufaufbaus mehrfach verschickt werden [TS-24.008, 2005, Kap. 5.5.6].

8: Erhält das MS ein `ALERTING` [TS-24.008, 2005, Kap. 5.2.1.5] vom Netzwerk, wurde der Anruf durchgestellt und es klingelt beim angerufenen Netzteilnehmer. Dem Anrufer wird das durch Abspielen des Freizeichens signalisiert.

9: Wenn der angerufene Netzteilnehmer abgehoben hat, wird dem MS das mit einer `CONNECT`-Nachricht mitgeteilt. Das MS bestätigt diese mit einer `CONNECT-ACKNOWLEDGE`-Nachricht und wechselt in den „call-active“ State. Die beiden Gesprächspartner sind jetzt erfolgreich miteinander verbunden und auf dem TCH werden ihre Sprachdaten übertragen.

Obiger Ablauf wurde mit der Aufzeichnung eines ausgehenden Anrufs mit Wireshark verifiziert. Der Anruf erfolgte von einem mit OsmocomBB gesteuerten Motorola C123 über eine reguläre E-Plus BTS. Der analysierte Mitschnitt wurde von OsmocomBB erstellt und kann im Anhang in Codebeispiel D.2 eingesehen werden.

Ergebnis:

Das Opfer kann im `CM-SERVICE-REQUEST` identifiziert werden, da in diesem seine Identität in Form der TMSI oder IMSI enthalten ist. Der `CM-SERVICE-REQUEST` ist Teil des MM und wird vom SABM huckepack getragen (piggybacking). Der SABM ist unverschlüsselt, die Nachrichten können somit vom MitM$_{Um}$ nach der Identität des Opfers gefiltert werden. Da die TMSI nur die temporäre Identität eines Netzteilnehmers ist, muss sie seiner eindeutigen Identität (IMSI oder MSISDN), zugeordnet werden. Dazu gibt es zwei Möglichkeiten:

- **Silent SMS**: Eine „lautlose“ SMS wird an die Telefonnummer des Opfers geschickt und der PCH des BTS, in dessen Zuständigkeitsbereich es sich befindet, auf Paging Nachrichten abgehört. Die TMSI, die über die eingehende SMS benachrichtigt wird, gehört zur MSISDN des Opfers.

- **IMSI Catcher**: Hat der Angreifer über ein MitM-Gerät Zugriff auf den Datenverkehr des Um, kann er auf alle eingehenden `L3-SERVICE-REQUESTS` mit einem `IDENTITY-REQUEST` die IMSI des Netzteilnehmers abfragen. Laut Spezifikation müssen MS jederzeit bereit sein, auf einen Identity Request zu antworten [TS-24.008, 2005, Kap. 4.3.3.2]. Hat die MS als Identität die TMSI im `L3-SERVICE-REQUEST` angegeben, bekommt der Angreifer so die Zuordnung zur IMSI.

Kennt der Angreifer die TMSI des Opfer, filtert er den SDCCH der BTS nach SABM-Nachrichten, die einen `CM-SERVICE-REQUEST` tragen. Diese sind immer noch unverschlüsselt, er kann also mit dem „CM Service Type“ Feld überprüfen, ob der angeforderte Dienst ein „CM mobile originated call establishment“ ist. Da nur ausgehende Anrufe für den Angriff von Interesse sind, können alle anderen angeforderten Dienste

verworfen werden.

Die Verschlüsselung der Signalisierung beginnt erst mit dem `CIPHERING-MODE-COMMAND`. Da dieser selbst noch unverschlüsselt übertragen wird, kann er identifiziert werden. Ab dem `CIPHERING-MODE-COMMAND` werden im Uplink die 184 Datenbit der LAPDm-Frames verschlüsselt. Der Angreifer hat keinen Zugriff mehr auf Informationen ab einschließlich Schicht 2 und kann deshalb die `SETUP`-Nachricht nicht ohne Weiteres identifizieren. Vom MS werden aber, wenn keine Fehler in der Datensicherungsschicht auftreten, nach dem `CIPHERING-MODE-COMMAND` immer die zwei gleichen Signalisierungsnachrichten gesendet. Zuerst ein LAPDm-RR-Frame, dann die `CIPHERING-MODE-COMPLETE`-Nachricht. Unter der Voraussetzung, dass auf Ebene 2 keine Fehler aufgetreten sind, kann der Angreifer die `SETUP`-Nachricht im verschlüsselten Datenstrom also durch Mitzählen aller Nachrichten ab dem `CIPHERING-MODE-COMMAND` lokalisieren.

5.4. Analyse des CM-Service-Request

Da der CM-Service-Request auf dem SDCCH übertragen wird, ist das verwendete LAPDm-Format A (siehe Abschnitt 3.3.3). Die Nachricht beginnt also mit 24 Bit LAPDm-Header Informationen, gefolgt von maximal 160 Bit Schicht 3 Header und/oder Nutzdaten. Da die Nachricht unverschlüsselt übertragen wird, hat ein Angreifer Zugriff auf alle übertragenen Informationen. Die Identifizierung der für den Angriff in Frage kommenden CM-Service-Request Nachrichten im MitM$_{Um}$ erfolgt in drei Schritten:

Schritt 1 – Filtern nach logischem Kanal

Die Konfiguration der physikalischen Kanäle einer BTS, also deren Multiframe Struktur, kann dem BCCH entnommen werden [TS-05.02, Kap. 3.3.2.3]. Der Angreifer kann damit den Datenstrom auf dem Downlink in logische Kanäle unterteilen und nach dem SDCCH filtern. Alle anderen logischen Kanäle müssen nicht betrachtet werden.

Schritt 2 – Filtern nach Schicht 2 Informationen

Die auf dem SDCCH übertragene Nachricht ist vom LAPDm-U-Format (siehe Abbildung 3.9). Das LAPDm-Nachrichtenformat wird durch Bit 1 und 2 des Kontrollfeldes beschrieben. Die Bitfolge `[1 1]` steht für das U-Format. Alle Nachrichten eines anderen Formats müssen nicht näher betrachtet werden. Das Kontrollfeld des U-Formats enthält mehrere U-Bits, die für die Beschreibung der verwendeten U-Funktion verwendet werden (siehe Abbildung 3.9). Die SABM-Funktion wird durch folgende Bitfolge beschrieben:

Bit	8	7	6	5	4	3	2	1
Bedeutung	Funktion			P/F	Funktion		Format	
SABM	0	0	1	P	1	1	1	1

Tabelle 5.1.: Der LAPDm-Header für die SABM-Funktion, nach [TS-04.06, 2008, Tabelle 4]

Wenn die U-Bits ein SABM beschreiben wird die Nachricht weiter betrachtet, ansonsten wird sie verworfen.

Schritt 3 – Filtern nach Schicht 3 Informationen

Als nächstes wird die Schicht 3 Information der Nachricht analysiert [TS-24.008, 2005, Kap. 9.2.9]. Der Aufbau eines CM-Service-Requests ist in Abbildung 5.7 dargestellt.

IEI	Information element	Type / Reference	Presence	Format	Length
	Mobility management protocol discriminator	Protocol discriminator 10.2	M	V	½
	Skip Indicator	Skip Indicator 10.3.1	M	V	½
	CM Service Request message type	Message type 10.4	M	V	1
	CM service type	CM service type 10.5.3.3	M	V	½
	Ciphering key sequence number	Ciphering key sequence number 10.5.1.2	M	V	½
	Mobile station classmark	Mobile station classmark 2 10.5.1.6	M	LV	4
	Mobile identity	Mobile identity 10.5.1.4	M	LV	2-9
8-	Priority	Priority Level 10.5.1.11	O	TV	1

Abbildung 5.7.: Der Aufbau des CM-Service-Request, aus [TS-24.008, 2005, Tabelle 9.2.11]

Die ersten drei Byte des Schicht 3 Headers sind immer gleich aufgebaut und ermöglichen die Unterscheidung der Schicht 3 Protokolle und Funktionen (siehe Abbildung 3.11). In obiger Tabelle mit M gekennzeichnete Elemente sind Pflichtfelder („mandatory“), O bedeutet optional. Die Elemente, die vom Angreifer untersucht werden müssen, haben folgende Reihenfolge:

- **Protocol Discriminator** == `0x5` == `[0 1 0 1]`, für MM (siehe Tabelle 3.2).
- **Message Type** == `0x24` == `[x x 1 0 0 1 0 0]`, für `CM-SERVICE-REQUEST` [TS-24.008, 2005, Tabelle 10.2]. Mit x gekennzeichnete Bits sind in MM anderweitig reserviert und gehen nicht in den Nachrichtentyp mit ein.
- **CM Service Type** == `0x1` == `[0 0 0 1]`, für „Mobile originating call establishment“ [TS-24.008, 2005, Tabelle 10.5.91]. Dieser Typ beschreibt einen ausgehenden Anruf.

Alle unpassenden Nachrichten können verworfen werden.

5.5. Analyse der Setup-Nachricht

Um den Anruf umzuleiten muss der Angreifer beim Aufbau des Telefongesprächs, die Telefonnummer des angerufenen Opfers (Bob) austauschen. Die in GSM dafür zuständige Nachricht ist die Setup-Nachricht des CC-Protokolls. Die Informationen der Setup-Nachricht sind kanalkodiert (siehe Abschnitt 3.5) und mit dem zwischen MS und BTS ausgehandelten A5 Verschlüsselungsverfahren geschützt (siehe Unterabschnitt 3.6.4). Ein Angreifer kann also nicht ohne Weiteres die Informationen auslesen, aufgrund des fehlenden Integritätsschutzes in Kombination mit Stromverschlüsselung jedoch bekannte Teile der Daten ändern (siehe Abschnitt 5.1). Um die Telefonnummer zu ändern, muss er deren Ziffern, Kodierung und Position in der Setup-Nachricht kennen. Diese ist in [TS-24.008, Kap. 9.3.23.2] spezifiziert, ihr Aufbau ist in Tabelle Abbildung 5.8 zu sehen. Das Feld in der die Telefonnummer steht ist rot markiert.

IEI	Information element	Type / Reference	Presence	Format	Length
	Call control protocol discriminator	Protocol discriminator 10.2	M	V	1/2
	Transaction identifier	Transaction identifier 10.3.2	M	V	1/2
	Setup message type	Message type 10.4	M	V	1
D-	BC repeat indicator	Repeat indicator 10.5.4.22	C	TV	1
04	Bearer capability 1	Bearer capability 10.5.4.5	M	TLV	3-16
04	Bearer capability 2	Bearer capability 10.5.4.5	O	TLV	3-16
1C	Facility(simple recall alignment)	Facility 10.5.4.15	O	TLV	2-
5D	Calling party sub-address	Calling party subaddr. 10.5.4.10	O	TLV	2-23
5E	Called party BCD number	Called party BCD num. 10.5.4.7	M	TLV	3-43
6D	Called party sub-address	Called party subaddr. 10.5.4.8	O	TLV	2-23
D-	LLC repeat indicator	Repeat indicator 10.5.4.22	O	TV	1
7C	Low layer compatibility I	Low layer comp. 10.5.4.18	O	TLV	2-18
7C	Low layer compatibility II	Low layer comp. 10.5.4.18	O	TLV	2-18
D-	HLC repeat indicator	Repeat indicator 10.5.4.22	O	TV	1
7D	High layer compatibility i	High layer comp. 10.5.4.16	O	TLV	2-5
7D	High layer compatibility ii	High layer comp. 10.5.4.16	O	TLV	2-5
7E	User-user	User-user 10.5.4.25	O	TLV	3-35
7F	SS version	SS version indicator 10.5.4.24	O	TLV	2-3
A1	CLIR suppression	CLIR suppression 10.5.4.11a	C	T	1
A2	CLIR invocation	CLIR invocation 10.5.4.11b	C	T	1
15	CC capabilities	Call Control Capabilities 10.5.4.5a	O	TLV	4
1D	Facility $(CCBS)$ (advanced recall alignment)	Facility 10.5.4.15	O	TLV	2-?
1B	Facility (recall alignment Not essential) $(CCBS)$	Facility 10.5.4.15	O	TLV	2-?
2D	Stream Identifier	Stream Identifier 10.5.4.28	O	TLV	3

Abbildung 5.8.: Der Aufbau der Setup-Nachricht, aus [TS-24.008, 2005, Tabelle 9.70a]

Die angerufene Telefonnummer steht im Feld „Called party BCD number“, auf den Aufbau des Feldes und die verwendete Binary Coded Decimal (BCD)-Kodierung wird in Abschnitt 5.6 eingegangen. In diesem Abschnitt interessieren vor allem die Elemente, die vor der Telefonnummer stehen und ihre Position in der Nachricht verschieben können. Alle

anderen Elemente sind grau markiert und werden hier nicht behandelt. Sie können unter den in obiger Tabelle angegebenen Referenzen in der TS-24.008 Spezifikation nachgelesen werden.

- **Protocol Discriminator**: Das Pflichtfeld ist Teil des Standardheaders von Schicht 3 und hat die feste Länge von 4 Bit. Er verschiebt die Telefonnummer um einen konstanten Wert und ist keine Gefahr für den Angriff. Für CC hat er den Wert `0x3 == [0 0 1 1]` (siehe Tabelle 3.2).

- **Transaction Identifier**: Das Pflichtfeld ist Teil des Standardheaders von Schicht 3 und hat die feste Länge von 4 Bit. In der Setup-Nachricht wird er nicht verwendet und muss mit Nullbits befüllt werden [TS-24.008, 2005, Kap. 8.3.1].

- **Message Type**: Der Typ einer Setup-Nachricht ist `0x5 == [x x 0 0 0 0 1 1]` [TS-24.008, 2005, Tabelle 10.3]. Mit x gekennzeichnete Bits gehen bei CC nicht in den Typen ein. Das Pflichtfeld hat eine feste Länge von einem Byte.

- **BC Repeat Indicator**: Dieses Feld ist optional und ist gesetzt, wenn beide „Bearer Capability“ Felder vorhanden sind.

- **Bearer Capability 1**: Das Feld ist optional und liefert der BTS Informationen über die Fähigkeiten des MS, was Sprachkodierung und Modus angeht. Ist das Feld nicht gesetzt, wird ein Default-Sprachmodus verwendet. Es hat eine dynamische Länge von bis zu 16 Byte.

- **Bearer Capability 2**: Das Feld ist optional und kommt nur in Kombination mit „BC Repeat Indicator“ vor. Das MS kann damit einen Fallback angeben, falls das Netzwerk die Angaben in „Bearer Capability 1“ nicht akzeptiert. Es hat eine variable Länge von bis zu 16 Byte.

- **Facility**: Das Feld ist optional und von variabler Länge [TS-24.080, 2011, Kap. 3.6]. Es kann verwendet werden, um anrufbezogene SS's anzufordern und findet bei Completion of Calls to Busy Subscriber (CCBS), bei einen vom Netzwerk initiierten, automatischen Rückruf des MS Verwendung. Für einen normalen, vom MS ausgehenden Anruf wird es nicht genutzt.

- **Calling Party Subaddress** Das Feld ist optional und hat eine variable Länge.

Die ISDN-Subaddresse ist in TREC-I.330 definiert, der Aufbau des TLV-Elements findet sich in TS-24.008, Abb. 10.5.94 und 10.5.95. Das Feld wird in der Regel in der Setup-Nachricht nicht angegeben, da der Anrufer bereits eindeutig mit seiner IMSI im Netz identifiziert ist und seine MSISDN dem VLR bekannt ist.

– **Called Party BCD Number** In diesem Pflichtfeld steht die BCD-kodierte MSISDN des angerufenen Netzteilnehmers. Das Feld kann zwischen drei und 43 Byte lang sein, was Platz für zwei bis 82 BCD-kodierte Ziffern entspricht. Die hohe maximale Länge kommt daher, dass das Feld nicht nur ISDN und MSISDN-Nummern, sondern auch Adressen anderen Typs enthalten kann.

Die Felder, die wegen ihrer variablen Länge für die „Known Plaintext“ Voraussetzung des Angriffs eine Gefahr darstellen, werden im Folgenden genauer betrachtet.

BC-Repeat-Indicator, Bearer-Capability-1 und Bearer-Capability-2

In den im Zuge dieser Masterarbeit aufgezeichneten Signalisierungsnachrichten wurde das Fallbackfeld Bearer-Capability-2 nicht im Anrufaufbau verwendet. Für die Arbeit wird deshalb angenommen, dass die Felder BC-Repeat-Indicator und Bearer-Capability-2 nicht in der Setup-Nachricht vorkommen.

Bearer-Capability-1 konnte in den Aufzeichnungen mehrfach beobachtet werden. Das Feld liefert dem BTS Präferenzen und Fähigkeiten des MS für die zu etablierende Sprachverbindung. Der Inhalt ist also abhängig von der im MS verbauten Hardware und damit dessen Hersteller und Modell. Der Angreifer kann entweder durch Social-Engineering oder die IMEI an Informationen über das vom Opfer verwendeten Mobilfunkgeräts kommen. Die IMEI kann wie die IMSI, mit einem Identity Request vom IMSI-Catcher, beziehungsweise dem MitM-Gerät in der Funkschnittstelle, abgefragt werden. Verschiedene Onlinedienste[1][2][3] liefern Informationen zur IMEI, wie Hersteller und Modell des Geräts. Es wird angenommen, dass das vom Opfer verwendete Mobilfunkgerät den Inhalt von Bearer-Capability-1 eindeutig festlegt, womit dessen Länge als bekannt vorausgesetzt werden kann.

Für die Arbeit wurden Mitschnitte der Modelle Motorola C123 und Samsung Galaxy

[1] `http://www.imei.info`
[2] `https://imeidata.net`
[3] `http://imei-number.com/imei-number-lookup`

S4 Active aufgezeichnet, für die die Annahmen gültig waren. Durch Tests mit mehreren, verschiedenen Geräten sollten sie aber noch klarer bestätigt werden.

Facility

Das Facility-Feld wird in der Setup-Nachricht nur in CCBS, für den Rückruf einen beschäftigten Netzteilnehmers beim Netzwerk verwendet. Das ist eine Routine, die es dem Netzwerk erlaubt, den Anrufs eines CCBS-fähigen MS zu initiieren [TS-24.008, 2005, Kap. 5.2.3]. Sie wird verwendet, falls das MS gerade beschäftigt ist und nicht sofort auf einen eingehenden Anruf antworten kann. Das MS erhält dabei vom Netzwerk das Facility-Feld mit Daten, die den nicht entgegengenommenen Anruf referenzieren. Beim anschließenden Anrufaufbau fügt das MS das dieses Facility-Feld in die Setup-Nachricht ein [TS-24.008, 2005, Kap. 5.2.3.2.1] wodurch das Netzwerk den Anruf zuordnen und vermitteln kann.

Der oben beschriebene Anwendungsfall trifft beim vorgestellten Angriff nicht zu. Das Opfer initiiert den Anruf von sich aus.

Calling party subaddress

Die ISDN-Subadresse wurde spezifiziert, um Routing von ISDN-Adressen im Open Systems Interconnection (OSI)-Adressierungsmodell zu ermöglichen. Mit der Subadresse kann die Netzwerkschicht der Adresse spezifiziert werden [TREC-I.334], [TREC-I.334]. Bei einem Anruf von einem GSM-Netzteilnehmer bei einem GSM-Netzteilnehmer wird das Feld nicht verwendet, da auf beiden Seiten die ISDN-Adressierung verwendet wird.

Für den Angriff kann das Feld also als leer angenommen werden.

5.6. Analyse der Telefonnummer

Der Aufbau einer MSISDN folgt dem TREC-E.164 Standard für ISDN-Adressierung und ist in TS-23.003, Abschnitt 3.3 definiert. Die MSISDN setzt sich aus Country Code (CC), Network Destination Code (NDC) und Subscriber Number (SN) zusammen und hat eine

maximale Länge von 15 Zeichen. Dabei adressiert CC das Land in dem der Netzteilnehmer angemeldet ist, NDC das Netzwerk und SN den Teilnehmer. Die Kombination MSISDN = CC || NDC || SN ist international eindeutig.

Für den Angriff muss die angerufene Nummer (von Bob) nicht vollständig bekannt sein. Die verwendeten Mobilfunknummern haben, wie am Beispiel der Deutschen Adaption des Standards (Tabelle 5.2) zu sehen ist, mehrere gleiche oder ähnliche Teile. Hat der Angreifer ein genügend großes Portfolio von MSISDNs, an die er den Anruf umleiten kann, reicht im Idealfall die Kenntnis und der Austausch einer einzelnen Ziffer. Die fehlenden Ziffern von Bob's Nummer können aus der Rufnummer, bei der das umgeleitete Telefonat eingeht, hergestellt werden. Wird in der Nummer ein internationales Präfix verwendet, könnte der Anruf vom Angreifer auch ins Ausland umgeleitet [TREC-E.164, 2010a] und von dort, entsprechende Kontakte des Angreifers vorausgesetzt, zu ihm weitergeleitet werden.

<table>
<tr><th>Präfix
(1-2 Ziffern)</th><th>Country Code
(0-3 Ziffern)</th><th colspan="3">nationale Rufnummer (10-11 Ziffern)</th></tr>
<tr><td>national
[0]</td><td>-</td><td rowspan="2">Dienste-
kennzahl
[15, 16, 17]
(2 Ziffern)</td><td colspan="2">Teilnehmerrufnummer (8-9 Ziffern)</td></tr>
<tr><td>international
[00]</td><td>1-3 Ziffern</td><td>Blockkennung
(3-5 Ziffern)</td><td>Endeinrichtungs-
nummer
(6-8 Ziffern)</td></tr>
</table>

Tabelle 5.2.: Der Aufbau deutscher Mobilfunkrufnummern, nach [Bundesnetzagentur, 2013]

Das Präfix einer internationalen Rufnummer ist 00, ihm folgt der CC des Landes vor der nationalen Telefonnummer. Das Präfix einer nationalen Nummer ist nur eine einzelne 0, ohne CC kommt direkt danach die nationale Rufnummer. Nationale Rufnummern werden nur innerhalb des Landes, in dem sich der Anrufer befindet, vermittelt, da sie international nicht eindeutig sind. In Deutschland entspricht die Dienstekennzahl dem NDC und die Teilnehmerrufnummer der SN.

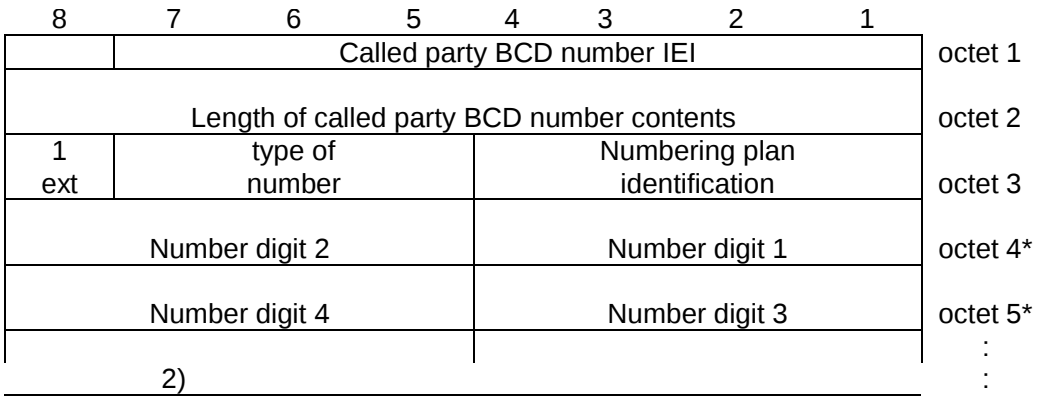


Abbildung 5.9.: Der Aufbau des Datenfelds für die angerufene Telefonnummer, aus [TS-24.008, 2005, Abb. 10.5.4.7]

Der Aufbau des Datenfelds der angerufenen Telefonnummer [TS-24.008, 10.5.4.7] ist in Abbildung 5.9 zu sehen. Das Feld ist ein TLV-Element, das erste Byte wird für den IEI und das zweite für die Länge der Daten verwendet. Bit 8 im dritten Oktett ist das TI Extension Bit [TS-24.007, Kap. 11.2.3.1.3], wird nicht genutzt und ist immer 1. Danach kommt der Typ der Nummer. Für die Typen national und international wird in der folgenden Rufnummer das Präfix weggelassen. Bei unbekanntem Typ ist das Präfix in der Telefonnummer enthalten und muss vom Netzwerk interpretiert werden. Die Kodierung der relevanten Typen ist in der folgenden Tabelle zusammengefasst. Im „Numbering Plan Identification“ Feld steht für Mobilfunknummern der Code `[0 0 0 1]` für eine nach „ISDN/telephony numbering plan (Rec. E.164/E.163)“ aufgebaute Nummer.

Bit 7	Bit 6	Bit 5	Beschreibung
0	0	0	unbekannt
0	0	1	international
0	1	0	national

Tabelle 5.3.: Die verschiedenen Rufnummerntypen, nach [TS-24.008, Tabelle 10.5.118]

Byte 4* und folgende werden im Datenfeld für die BCD-kodierten Ziffern der Telefonnummer verwendet. Dabei muss die Reihenfolge beachtet werden, die entgegengesetzt der Leserichtung ist. Da die kodierte Rufnummer byte-aligned ist, bleiben bei ungeradzahligen Rufnummern vier Bit übrig, die mit einer Endmarkierung als ungenutzt gekennzeichnet werden. Die in GSM verwendete Variante der BCD-Kodierung ist in folgender Tabelle dargestellt.

Ziffer/Zeichen	binäre Darstellung
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
*	1010
#	1011
a	1100
b	1101
c	1110
Endmarkierung	1111

Tabelle 5.4.: Die Adaption der BCD-Kodierung in GSM, nach [TS-24.008, Tabelle 10.5.118]

In Codebeispiel 5.1 wird ein mit Beispieldaten gefülltes Called-Party-Number-Feld analysiert und in seine Bestandteile zerlegt. Die Daten wurden aus dem Wireshark Mitschnitt zu Codebeispiel D.3 entnommen und mithilfe von Wireshark und der Spezifikation in TS-24.008 interpretiert.

```
Kodiertes Feld in Hexdarstellung:
5e 09 81 00 94 71 15 18 18 18 f8

Bedeutung:
5e == [0101 1100] := T, Typ, IEI
09 == [0000 1001] := L, Länge des Datenteils in Bytes
81 == [1000 0001] := [Extension][Number Type][Numbering Plan]
      [1        ] := keine Erweiterung
      [ 000     ] := unbekannter Nummerntyp
      [     0001] := ISDN Nummernplan nach ITU.T Rec. E.164/E.163
00 == [0000 0000] := 00 (internationales Prefix)
94 == [1001 0100] := 49 (Country Code Deutschland)
```

```
71 == [1001 0100] := 17 (nationale Vorwahl)
15 == [0001 0101]
      [     0101] := 5  (nationale Vorwahl)
      [0001     ] := 1  (nationale Rufnummer)
18 == [0001 1000] := 81 (nationale Rufnummer)
18 == [0001 1000] := 81 (nationale Rufnummer)
18 == [0001 1000] := 81 (nationale Rufnummer)
f8 == [1111 1000]
      [     1000] := 8  (nationale Rufnummer)
      [1111     ] := Endmarkierung wegen ungerader Ziffernzahl

Dekodierte Rufnummer:
00 49 175 18181818
```

Codebeispiel 5.1: Dekodierung und Inhalt des Called Party Number Feldes an einem Beispiel, Daten des Feldes entnommen aus Wireshark Mitschnitt, siehe Codebeispiel D.3

5.7. Mathematischer Nachweis der Manipulation einer kodierten, verschlüsselten Setup-Nachricht

Der Angreifer hat die verschlüsselte und kodierte Setup-Nachricht als Eingangparameter für die Manipulation. Um die Manipulation mathematisch darstellen zu können, werden in 5.1 Variablen definiert.

$$
\begin{aligned}
&\mathbb{B} && := \{0,1\}, \text{Menge der binären Zahlen} \\
&\oplus && , \text{binäre Addition oder bitweises exklusives ODER (XOR)} \\
&\odot && , \text{binäre Multiplikation oder bitweises UND} \\
&\boldsymbol{d} \in \mathbb{B}^{184} && , \text{Daten im Klartext} \\
&\boldsymbol{d_m} \in \mathbb{B}^{184} && , \text{manipulierte Daten im Klartext} \\
&\boldsymbol{r} \in \mathbb{B}^{228} && , \text{Rest der Polynomdivision eines Blockcodes} \\
&\boldsymbol{k_s} \in \mathbb{B}^{456} && , \text{Schlüsselstrom der Stromverschlüsselung}
\end{aligned}
\tag{5.1}
$$

5.2 definiert die bei der Kanalkodierung eingesetzten Verfahren auf den Datenströmen als Funktionen auf binären Vektoren.

$$\begin{aligned}
&\boldsymbol{u} : \mathbb{B}^{184} \to \mathbb{B}^{228}, \boldsymbol{x} \mapsto \boldsymbol{u}(\boldsymbol{x}) && \text{, Funktion Blockkodierung} \\
&\boldsymbol{c} : \mathbb{B}^{228} \to \mathbb{B}^{456}, \boldsymbol{x} \mapsto \boldsymbol{c}(\boldsymbol{x}) && \text{, Funktion Faltungskodierung} \\
&\boldsymbol{i} : \mathbb{B}^{456} \to \mathbb{B}^{456}, \boldsymbol{x} \mapsto \boldsymbol{i}(\boldsymbol{x}) && \text{, Funktion Interleaving} \\
&\boldsymbol{a} : \mathbb{B}^{456} \to \mathbb{B}^{456}, \boldsymbol{x} \mapsto \boldsymbol{x} \oplus \boldsymbol{k_s} && \text{, Funktion Verschlüsselung} \\
&\boldsymbol{b} : \mathbb{B}^{456} \to \mathbb{B}^{464}, \boldsymbol{x} \mapsto \boldsymbol{b}(\boldsymbol{x}) && \text{, Funktion Burstmapping} \\
&\boldsymbol{u_r}(\boldsymbol{x}) = \boldsymbol{u}(\boldsymbol{x}) \oplus \boldsymbol{r} && \text{, Blockcode mit definiertem Rest} \\
&(\boldsymbol{b} \circ \boldsymbol{a} \circ \boldsymbol{i} \circ \boldsymbol{c} \circ \boldsymbol{u_r})(\boldsymbol{d}) && \text{, kanalkodierte, verschlüsselte LAPDm Daten}
\end{aligned} \tag{5.2}$$

Für Lineare Abbildungen $\boldsymbol{f} : \boldsymbol{V} \to \boldsymbol{W}$ auf dem Galoiskörper $\mathbb{F}_2$ 2. Ordnung gilt [Werner, 2008, S. 142 Tabelle 3.2]:

$$\boldsymbol{f}(a \odot \boldsymbol{x} \oplus \boldsymbol{y}) = a \odot \boldsymbol{f}(\boldsymbol{x}) \oplus \boldsymbol{f}(\boldsymbol{y}), \text{ mit } \boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{V} \text{ und } a \in \{0, 1\} \tag{5.3}$$

Die XOR Operation ist kommutativ, assoziativ und hat als inverse Element das Element selbst und als neutrales Element 0.

$$\boldsymbol{x} \oplus \boldsymbol{x} = 0 \qquad \text{inverses Element} \tag{5.4}$$

$$\boldsymbol{x} \oplus 0 = \boldsymbol{x} \qquad \text{neutrales Element} \tag{5.5}$$

$$\boldsymbol{x} \oplus \boldsymbol{y} = \boldsymbol{y} \oplus \boldsymbol{x} \qquad \text{Kommutativität} \tag{5.6}$$

$$(\boldsymbol{x} \oplus \boldsymbol{y}) \oplus \boldsymbol{z} = \boldsymbol{x} \oplus (\boldsymbol{y} \oplus \boldsymbol{z}) \qquad \text{Assoziativität} \tag{5.7}$$

Da Blockkodierung, Faltungskodierung und Interleaving lineare Abbildungen sind [Werner, 2008, S. 142 ff.], gilt für sie Additivität.

$$\begin{aligned}
&\boldsymbol{u}(\boldsymbol{x} \oplus \boldsymbol{y}) = \boldsymbol{u}(\boldsymbol{x}) \oplus \boldsymbol{u}(\boldsymbol{y}) \\
&\boldsymbol{c}(\boldsymbol{x} \oplus \boldsymbol{y}) = \boldsymbol{c}(\boldsymbol{x}) \oplus \boldsymbol{c}(\boldsymbol{y}) \\
&\boldsymbol{i}(\boldsymbol{x} \oplus \boldsymbol{y}) = \boldsymbol{i}(\boldsymbol{y}) \oplus \boldsymbol{i}(\boldsymbol{y})
\end{aligned} \tag{5.8}$$

In GSM wird für die Blockkodierung mit dem Firecode ein Rest $\boldsymbol{r}$ definiert, der bei der Polynomdivision übrig bleiben muss (siehe Unterabschnitt 3.5.1. Für die Funktion $\boldsymbol{u_r}(\boldsymbol{x})$

aus 5.1 ändert sich die Additivität damit wie folgt:

$$\boldsymbol{u}_r(\boldsymbol{x}) \oplus \boldsymbol{u}_r(\boldsymbol{y}) = \boldsymbol{u}(\boldsymbol{x}) \oplus \boldsymbol{r} \oplus \boldsymbol{u}(\boldsymbol{y}) \oplus \boldsymbol{r} \overset{5.4}{=} \boldsymbol{u}(\boldsymbol{x}) \oplus \boldsymbol{u}(\boldsymbol{y}) \overset{5.8}{=} \boldsymbol{u}(\boldsymbol{x} \oplus \boldsymbol{y}) \tag{5.9}$$

$$\Rightarrow \boldsymbol{u}_r(\boldsymbol{x} \oplus \boldsymbol{y}) = \boldsymbol{u}(\boldsymbol{x} \oplus \boldsymbol{y}) \oplus \boldsymbol{r} \overset{5.9}{=} \boldsymbol{u}_r(\boldsymbol{x}) \oplus \boldsymbol{u}_r(\boldsymbol{y}) \oplus \boldsymbol{r} \tag{5.10}$$

Für modifizierte Daten $\boldsymbol{d}_m$ gilt somit:

$$\begin{aligned} (\boldsymbol{c} \circ \boldsymbol{u}_r)(\boldsymbol{d}_m) &\overset{5.4,5.5}{=} (\boldsymbol{c} \circ \boldsymbol{u}_r)((\boldsymbol{d} \oplus \boldsymbol{d}) \oplus \boldsymbol{d}_m) \\ &\overset{5.6}{=} (\boldsymbol{c} \circ \boldsymbol{u}_r)(\boldsymbol{d} \oplus (\boldsymbol{d} \oplus \boldsymbol{d}_m)) \\ &\overset{5.10}{=} \boldsymbol{c}(\boldsymbol{u}_r(\boldsymbol{d}) \oplus \boldsymbol{u}_r(\boldsymbol{d} \oplus \boldsymbol{d}_m) \oplus \boldsymbol{r}) \\ &\overset{5.8}{=} (\boldsymbol{c} \circ \boldsymbol{u}_r)(\boldsymbol{d}) \oplus (\boldsymbol{c} \circ \boldsymbol{u}_r)(\boldsymbol{d} \oplus \boldsymbol{d}_m) \oplus \boldsymbol{c}(\boldsymbol{r}) \end{aligned} \tag{5.11}$$

Die Stromverschlüsselung kann als XOR Verknüpfung eines Datenstroms mit dem Schlüsselstrom $\boldsymbol{k}_s$ dargestellt werden. Wegen der Kommutativität und Assoziativität der XOR Operation gilt:

$$\begin{aligned} \boldsymbol{a}(\boldsymbol{x} \oplus \boldsymbol{y}) &= (\boldsymbol{x} \oplus \boldsymbol{y}) \oplus \boldsymbol{k}_s \\ &\overset{5.6}{=} \boldsymbol{x} \oplus (\boldsymbol{y} \oplus \boldsymbol{k}_s) = \boldsymbol{x} \oplus \boldsymbol{a}(\boldsymbol{y}) \\ &\overset{5.7}{=} \boldsymbol{y} \oplus (\boldsymbol{x} \oplus \boldsymbol{k}_s) = \boldsymbol{y} \oplus \boldsymbol{a}(\boldsymbol{x}) \end{aligned} \tag{5.12}$$

Mit Gleichung 5.12 können die verschlüsselten, kodierten Daten wie folgt dargestellt werden:

$$\begin{aligned} (\boldsymbol{a} \circ \boldsymbol{i} \circ \boldsymbol{c} \circ \boldsymbol{u}_r)(\boldsymbol{d}_m) &= (\boldsymbol{i} \circ \boldsymbol{c} \circ \boldsymbol{u}_r)(\boldsymbol{d}_m) \oplus \boldsymbol{k}_s \\ &\overset{5.11}{=} \boldsymbol{i}((\boldsymbol{c} \circ \boldsymbol{u}_r)(\boldsymbol{d}) \oplus (\boldsymbol{c} \circ \boldsymbol{u}_r)(\boldsymbol{d}_m \oplus \boldsymbol{d}) \oplus \boldsymbol{c}(\boldsymbol{r})) \oplus \boldsymbol{k}_s \\ &\overset{5.8}{=} (\boldsymbol{i} \circ \boldsymbol{c} \circ \boldsymbol{u}_r)(\boldsymbol{d}) \oplus (\boldsymbol{i} \circ \boldsymbol{c} \circ \boldsymbol{u}_r)(\boldsymbol{d}_m \oplus \boldsymbol{d}) \oplus (\boldsymbol{i} \circ \boldsymbol{c})(\boldsymbol{r}) \oplus \boldsymbol{k}_s \\ &\overset{5.12}{=} \underbrace{(\boldsymbol{a} \circ \boldsymbol{i} \circ \boldsymbol{c} \circ \boldsymbol{u}_r)(\boldsymbol{d})}_{\text{Teil 1}} \oplus \underbrace{(\boldsymbol{i} \circ \boldsymbol{c} \circ \boldsymbol{u}_r)(\boldsymbol{d}_m \oplus \boldsymbol{d})}_{\text{Teil 2}} \oplus \underbrace{(\boldsymbol{i} \circ \boldsymbol{c})(\boldsymbol{r})}_{\text{Teil 3}} \end{aligned} \tag{5.13}$$

Teil 1 der obigen Gleichung ist bekannt. Er kann mit der Umkehrfunktion der Burstmapping Funktion $\boldsymbol{b}^{-1}(\boldsymbol{x})$, aus der als bekannt vorausgesetzten, kanalkodierten und verschlüsselten LAPDm-Nachricht berechnet werden. Die Umkehrfunktion entspricht dem Burst unmapping beim Dekodierungsvorgang. Gleichung 5.14 zeigt den Zusammenhang.

$$(\boldsymbol{a} \circ \boldsymbol{i} \circ \boldsymbol{c} \circ \boldsymbol{u}_r)(\boldsymbol{d}) = \left(\left(\boldsymbol{b}^{-1} \circ \boldsymbol{b}\right) \circ \boldsymbol{a} \circ \boldsymbol{i} \circ \boldsymbol{c} \circ \boldsymbol{u}_r\right)(\boldsymbol{d}) \tag{5.14}$$

Teil 2 aus Gleichung 5.13 kann berechnet werden, da der manipulierte Teil der Originaldaten $\boldsymbol{d}$, die Daten durch die man diesen Teil ersetzen möchte $\boldsymbol{d_m}$, sowie Algorithmen für Interleaving, Faltungskodierung und Blockcode bekannt sind.

Teil 3 aus Gleichung 5.13 kann berechnet werden, da der Sollrest $\boldsymbol{r}$ des Blockcodes (siehe Unterabschnitt 3.5.1) sowie Algorithmen für Faltungskodierung und Interleaving bekannt sind.

Als Ergebnis kann festgehalten werden, dass ein Angreifer die Rufnummer des angerufenen Netzteilnehmers in der verschlüsselten, kodierten Nachricht manipulieren kann sofern er deren Inhalt und Position im Datenstrom kennt („Known Plaintext“). Diese Manipulation, in Kombination mit dem fehlenden Integritätsschutz in GSM, ist die Grundlage des vorgestellten Angriffs. Sie wird ausgenutzt, um den Anruf umzuleiten.

Die Manipulation wurde durch Anwendung der Algorithmen auf einen konkreten, verschlüsselten und kodierten Beispieldatensatz verifiziert. In Abschnitt C.3 kann das Ergebnis eingesehen werden.

6. Die praktische Umsetzung des Angriffs

Dieses Kapitel beschreibt die Implementierungen und Anwendungen, die im Zuge der Arbeit entstanden sind. Insbesondere wird auf die Programme eingegangen, mit denen die Machbarkeit des Angriffs praktisch verifiziert wurde. Zuerst wurde die Manipulation der Telefonnummer in einer verschlüsselten und kodierten Setup-Nachricht umgesetzt. Im Anschluss wurde der Angriff von der Identifikation des Opfers, über die Identifikation der Setup-Nachricht, bis hin zu deren Manipulation in einer Testumgebung mit virtualisierter Funkschnittstelle durchgeführt. Die Beschreibung der Einrichtung und Konfiguration der verwendeten Testumgebung ist im Anhang in Abschnitt C.1 zu finden.

6.1. Die Virtualisierung der physikalischen Schicht der Um-Schnittstelle

Für die Verifikation des MitM-Angriffs wurde die physikalische Schicht der Um-Schnittstelle virtualisiert. Damit ist eine Kommunikation über das Um möglich, ohne spezielle Transceiverhardware für MS, BTS und MitM zu benötigen. Die Verwendung einer Testumgebung mit virtueller Um-Schnittstelle hat mehrere Vorteile. Da für den Nachrichtenaustausch keine Transceiverhardware benötigt wird, kann deren Konfiguration und Implementierung als Fehlerquelle bei der Entwicklung ausgeschlossen werden. Datenflüsse können schnell und kostengünstig aufgezeichnet und analysiert werden, was bei der Softwareentwicklung die Überprüfung der Auswirkungen einer Änderung vereinfacht. Im Vergleich mit einer Testumgebung mit realer Funkschnittstelle kann die virtualisierte Testumgebung einfacher eingerichtet und konfiguriert werden. Zudem werden Probleme mit der Legalität einer BTS und dem Funkverkehr auf für GSM reservierten Frequenzbändern vermieden.

Als Grundlage für die Implementierung kamen zwei mögliche Ansätze in Frage. Die in MATLAB[1] geschriebene Virtualisierung der Funkschnittstelle von kkro[2] und die angefangene Implementierung einer virtuellen BTS im osmoBTS Projekt von Harald Welte[3]. Für die Masterarbeit wurde der zweite Ansatz verfolgt. Für die virtuelle BTS fehlten zwar noch Teile der Funktionalität und das Gegenstück in osmocomBB, die Implementierung war aber deutlich aktueller. Des Weiteren konnte so die vorhandene Funktionalität aus osmocomBB, osmoBTS und der libosmocore Bibliothek genutzt werden.

Um die Übertragung auf der Funkschnittstelle nachzubilden, müssen Nachrichten die TDMA/FDMA Informationen der physikalischen Schicht des Um enthalten (siehe Unterabschnitt B.2.1). Die **Frequenz** wird in GSM über die ARFCN festgelegt. Eine ARFCN definiert eine Uplink- und eine Downlinkfrequenz, über die MS und BTS kommunizieren. Der TDMA-Zeitschlitz legt den **physikalischen Kanal** der Nachricht fest, die FN ihren **logischen Kanal** (siehe Unterabschnitt 3.3.1). Durch physikalischen und logischen Kanal wird eindeutig festgelegt, wie die physikalische Schicht mit der Nachricht umgehen muss und über welchen SAP sie diese an höhere Schichten weiterleitet. Auf dem Um werden in der Regel auch Messungen der Signalqualität durchgeführt und übertragen. Für das virtuelle Um sind diese weniger von Bedeutung, da alle Nachrichten mit gleicher Qualität übertragen werden.

Für die Virtualisierung sollen die Nachrichten des Um statt über Funksignale zwischen Sockets auf einer Linux Maschine übertragen werden. Für die Kapselung von Um-Nachrichten und deren Übertragung über UDP/IP bietet die libosmocore Bibliothek den GSM Test Access Protocol (GSMTAP)-Header an. Test Access Protocol (TAP) Header sind dazu gedacht, einem Analyse- oder Testgerät Informationen über den Datenverkehr einer Netzwerkschnittstelle zugänglich zu machen. Er übernimmt in Osmocom Projekten die Funktion des Wireshark-Dissektors bei der Aufzeichnung und Protokollierung des Nachrichtenverkehr auf dem Um mit Wireshark. Von der Internet Assigned Numbers Authority (IANA) wurde dem GSMTAP-Dissektor der Port 4729 zugewiesen. Codebeispiel 6.1 zeigt den Aufbau des GSMTAP Headers.

```
/*! \brief Structure of the GTMTAP pseudo-header */
struct gsmtap_hdr {
uint8_t version;    /*!< version, set to 0x01 currently */
```

[1] https://de.mathworks.com
[2] https://github.com/kkroo/matphy
[3] https://github.com/osmocom/osmo-bts.git

```
uint8_t hdr_len;    /*!< length in number of 32bit words */
uint8_t type;       /*!< see GSMTAP_TYPE_* */
uint8_t timeslot;   /*!< timeslot (0..7 on Um) */

uint16_t arfcn;     /*!< ARFCN (frequency) */
int8_t signal_dbm;  /*!< signal level in dBm */
int8_t snr_db;      /*!< signal/noise ratio in dB */

uint32_t frame_number;  /*!< GSM Frame Number (FN) */

uint8_t sub_type;   /*!< Type of burst/channel, see above */
uint8_t antenna_nr; /*!< Antenna Number */
uint8_t sub_slot;   /*!< sub-slot within timeslot */
uint8_t res;        /*!< reserved for future use (RFU) */
} __attribute__((packed));
```

Codebeispiel 6.1: Der Aufbau des GSMTAP-Headers, aus der libosmocore Bibliothek

Die Implementierung des virtuellen Um auf UDP/IP Basis verwendet den GSMTAP-Header für die Datenübertragung und erweitert damit seinen bisherigen Einsatzbereich. Das Feld `arfcn` enthält ein Uplink Flag und definiert so eindeutig die **Frequenz**. Der **physikalische Kanal** wird von `timeslot` definiert und der **logische Kanal** von `frame_number` und `sub_type`. Im Gegensatz zur Funkschnittstelle werden die Pakete auf dem virtuellen Um dabei nicht in Bursts unterteilt. Es werden vollständige LAPDm-Nachrichten von bis zu 184 Bit in einer GSMTAP-Nachricht verpackt und über UDP/IP zur Gegenseite des virtuellen Um geschickt. Durch die Verwendung des GSMTAP-Headers erhält Wireshark außerdem out-of-the-box die Fähigkeit, den Datenverkehr auf dem virtuellen Um darzustellen. Der GSMTAP-Header ist nicht für die Übertragung von kanalkodierten und verschlüsselten Nachrichten konzipiert. Das virtuelle Um unterstützt deshalb weder Kanalkodierung noch verschlüsselte Verbindungen. Für die praktische Durchführung des Angriffs wurde diese Funktionalität im osmoMITM Projekt implementiert (siehe Abschnitt 6.5).

Wegen der benötigten Broadcast Eigenschaften der Funkschnittstelle, eignen sich Linux Multicast-Sockets für die Kommunikation auf dem virtuellen Um. Sie ermöglichen es einem BTS-Prozess, Nachrichten an mehrere MS-Prozesse zu versenden und umgekehrt. Für Multicast-Sockets ist auf Linux/Unix Systemen der IP-Adressraum von 224.0.0 - 240.0.0.0 reserviert. Es gibt Multicast-Server-Sockets und Client-Sockets. Ein Server versendet Nachrichten an eine IP-Adresse aus dem für Multicast reservierten Adressraum. Client-Sockets können diese empfangen, indem sie der Multicast Gruppe beitreten, die dieser

IP-Adresse entspricht. Die Konfiguration von Multicast kann dem Linux Benutzerhandbuch unter ip(7)[4] entnommen werden. Um die Funkschnittstelle mit Multicast-Sockets zu simulieren, wird für Uplink und Downlink je ein Client und ein Server-Socket benötigt. Abbildung 6.1 veranschaulicht die Verwendung der Multicast-Sockets im virtuellen Um.

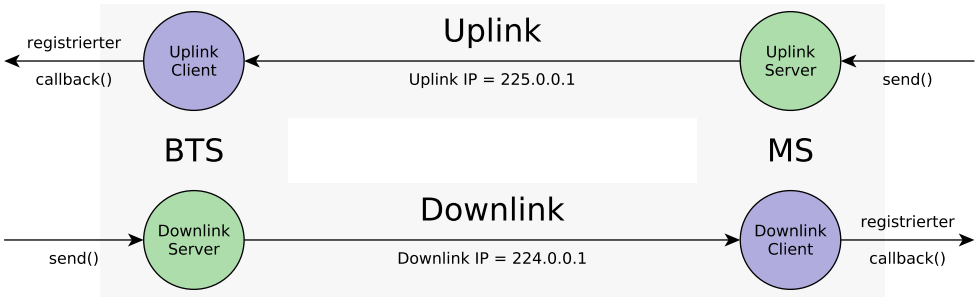


Abbildung 6.1.: Die Umsetzung der virtuellen Funkschnittstelle mit Multicast-Sockets, erstellt mit yEd

Wie in Abbildung 6.1 zu sehen, werden dabei je ein Server-Socket und ein Client-Socket zu Uplink und Downlink zusammengeschlossen. Die MS kann über die `send()` Funktion Daten auf dem Uplink senden und über den `callback()` auf dem Downlink empfangen. Der Callback wird beim Start des virtuellen Um registriert und ist die Methode, die vom Betriebssystem bei eingehenden Daten auf dem Socket aufgerufen wird. Umgekehrt dazu empfängt die BTS auf dem Uplink und sendet auf dem Downlink. Durch die Realisierung mit Multicast-Sockets, können mehrere MS und auch BTS am virtuellen Um angemeldet werden. Fügt man in Abbildung 6.1 zum Beispiel eine weitere MS hinzu und konfiguriert sie mit den selben IP-Adressen für Uplink (225.0.0.1) und Downlink (224.0.0.1), werden alle vom BTS gesendeten Nachrichten von beiden MS empfangen. Für die Virtualisierung wurde in den beiden Osmocom Projekten osmocomBB und osmoBTS die Implementierung der physikalischen Schicht der Funkschnittstelle ersetzt. Da auf der Socket-Ebene die Anforderungen des virtuellen Um auf MS und BTS-Seite die selben sind, wurde die damit zusammenhängende Funktionalität gekapselt. Die Datei `virt_um.c` wird auf beiden Seiten als geteilte Ressource verwendet, automatisiert die Erstellung der Sockets, lässt sich mit Uplink und Downlink-Adressen sowie der `callback()` Methode konfigurieren und bietet die `send()` Funktion an.

Mit der Einbindung des virtuellen Um in osmocomBB, osmoBTS und osmoMITM konnte

[4] `http://man7.org/linux/man-pages/man7/ip.7.html`

der MitM-Angriff erfolgreich simuliert und verifiziert werden. Auf der OsmoDevCon2017[5] wurde die Möglichkeit angesprochen, das virtuelle Um in Zukunft in das Projekt osmoTester einzubinden. Da dadurch schnelle, hardwareunabhängige Tests möglich sind, kann es von Jenkins [6] zur Überprüfung korrekter Nachrichtenabläufe auf der Funkschnittstelle bei neuen Commits verwendet werden. Jenkins ist ein webbasiertes Tool für die Automatisierung des Software-Building Prozesses und wird dafür in Osmocom verwendet.

6.2. OsmoBTS mit virtueller Um-Schnittstelle

Das osmoBTS Projekt unterstützt unterschiedliche Hardware, was sich in der Ordnerstruktur widerspiegelt. So ist ein Großteil der BTS-Funktionalität in einem Ordner `src/common`, der gemeinsam genutzte Funktionalität beinhaltet, ausgelagert. Er enthält vor allem die Implementierungen der hardwareunabhängigen, höheren Protokollschichten. Die Schnittstellen für die Unterstützung unterschiedlicher Transceiverhardware sind in mehreren weiteren Ordnern zu finden. Im Ordner `src/osmo-bts-trx` liegt zum Beispiel Code der Bursts auf einer UDP-Schnittstelle von verschiedenen SDR-Geräten entgegennimmt und verarbeitet. In `src/osmo-bts-sysmo` liegt eine Implementierung der physikalischen Schicht, mit der von sysmocom hergestellte BTS Produkte über Ethernet angebunden werden können. Der Automake[7] Build-Prozess verbindet die Implementierungen der Protokollschichten und baut für jede der unterstützen Hardwaregeräte eine eigene Anwendung zusammen.

Für osmoBTS wurde in dieser Arbeit im Ordner `src/osmo-bts-virtual` die physikalische Schicht zur Unterstützung der virtuellen Um-Schnittstelle implementiert. Sie erwartet und verarbeitet in GSMTAP verpackte Nachrichten auf einem Multicast-Socket. Folgende Grafik zeigt die Schnittstellen der virtuellen BTS (`osmo-bts-virt`) im Vergleich zu den anderen erwähnten BTS-Anwendungen `osmo-bts-trx` und `osmo-bts-sysmo`.

[5] `https://osmocom.org/projects/osmo-dev-con/wiki/OsmoDevCon2017`
[6] `https://jenkins.io/`
[7] `https://www.gnu.org/software/automake/manual`

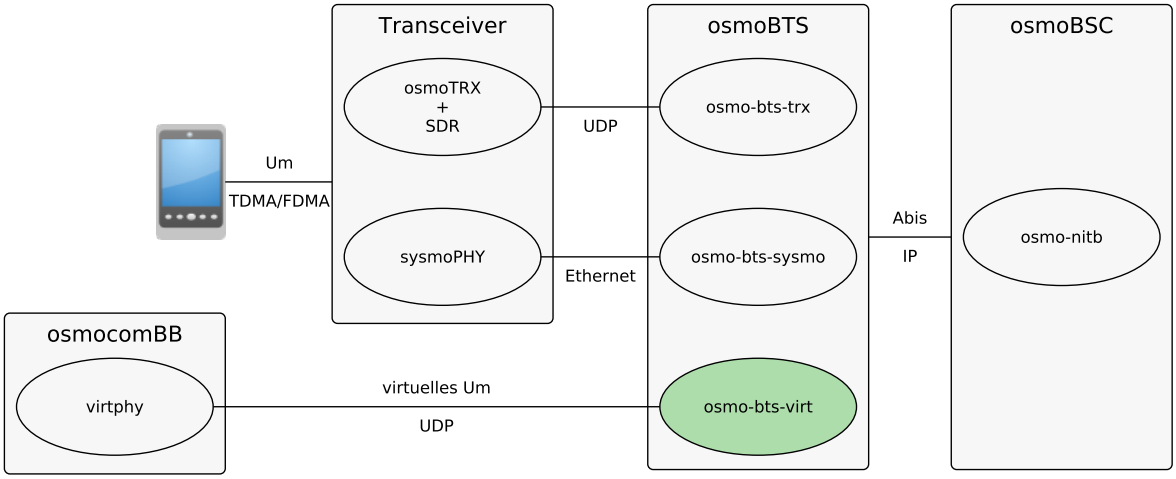


Abbildung 6.2.: Die Integration der virtuellen physikalischen Schicht in osmoBTS, erstellt mit yEd

Wie in obiger Grafik zu sehen, kann die virtuelle BTS direkt über das virtuelle Um mit verbundenen Mobiltelefonen kommunizieren und braucht keine Schnittstelle zu Transceiverhardware, wie `osmo-bts-trx` und `osmo-bts-sysmo`. Die Abis-Schnittstelle zum BSC genauso wie die Protokollschichten 2 und 3 sind im `common` Teil des osmoBTS Projekts implementiert und werden aus diesem eingebunden. In der virtuellen Testumgebung wurde in dieser Arbeit `osmo-nitb` als BSC verwendet (siehe Abschnitt 4.3). Neben der Funktion als BSC enthält `osmo-nitb` weitere Funktionalität von Komponenten des SS7 Netzwerks, wie zum Beispiel dem HLR. Bei der Implementierung der physikalischen Schicht von `osmo-bts-virt` wurden folgende Aufgaben gelöst:

Build-Prozess Das neue Programm wurde in den Automake Build-Prozess von osmoBTS eingebunden.

Implementierung des SAP der physikalischen Schicht zu höheren Schichten. Die in `src/common` implementierten Protokolle LAPDm und RR, rufen verschiedene SAP-Routinen der physikalischen Schicht auf (siehe Unterabschnitt 3.3.2). Sie erwarten vom SAP die korrekte Bestätigung (CON-Primitives) ihrer Anfragen (REQ-Primitives) und den Versand einiger IND-Primitives von der physikalischen Schicht. Da andernfalls die höheren Schichten nicht korrekt arbeiten, wurde in der physikalischen Schicht die vollständige SAP-Schnittstelle implementiert. Die Funktionalität der SAP-Routinen wurde an das virtuelle Um angepasst. Die Routine zur Übertragung einer Nutz- oder Signalisierungsnachricht verpackt

diese zum Beispiel in GSMTAP und schickt sie mit der `send()` Funktion von `virt_um.c` an die konfigurierte Multicast-Adresse. Um die RR-Schicht regelmäßig mit der aktuellen GSM-Zeit zu versorgen, wurde ein alle 4.615 ms (Dauer eines Frames, siehe Unterabschnitt B.2.1) wiederkehrender Timerinterrupt definiert. Die Interrupt-Routine signalisiert der RR-Schicht mit einem IND-Primitive den Anfang eines neuen Frames. Die Funktionalität einiger SAP-Routinen wird im virtuellen Um nicht simuliert und wurde deshalb nicht implementiert. Dazu gehören zum Beispiel „Frequency Hopping“, die Anpassung der Stärke des gesendeten Signals und die Verschlüsselung von RR-Verbindungen.

Parsen der eingehenden GSMTAP-Nachrichten. Da die Verwendung von GSMTAP zur Nachrichtenübertragung innerhalb von Osmocom Programmen neu ist, fehlte die Logik zum Auslesen der Informationen des Headers. Diese wurde implementiert.

Multiframe und TDMA-Scheduling. Damit ausgehende Nachrichten auf dem richtigen physikalischen und logischen Kanal gesendet werden, ist ein Scheduler notwendig. Es müssen periodische Aufgaben, wie das Broadcasting von SI-Nachrichten und einmalige Aufgaben, wie der durch einen SAP-Befehl initiierte Versand einer Nachricht, vom Scheduler für eine bestimmte FN geplant und ausgeführt werden. In osmoBTS ist eine Kombination aus Multiframe- und TDMA-Scheduler implementiert, dessen Basisfunktionalität im `common`-Teil zu finden ist. Die Funktionalität für die Planung von einmaligen und periodischen Aufgaben konnte übernommen werden. Die Routinen zur Ausführung, also in der Regel dem Versand einer Nachricht auf einem logischen Kanal, wurden neu geschrieben, so dass sie das virtuelle Um für die Übertragung nutzen.

Konfiguration der BTS über VTY. Um die virtuelle BTS über das Terminal konfigurieren zu können, wurde die VTY-Schnittstelle implementiert. Die Konfiguration wird im physikalischen Modell der virtuellen BTS (ein C-Struct) gespeichert und verschiedenen Routinen durch Shared-Memory zur Verfügung gestellt. Mit VTY können zum Beispiel die Multicast-Adressen für Uplink und Downlink konfiguriert werden.

Implementierung von OML. Beim Start meldet sich die BTS bei ihrem zuständigen BSC und nimmt anschließend auf der Abis Schnittstelle über den OML mehrere Konfigurationen von diesem entgegen. So kann der BSC zum Beispiel die Transceiver-

hardware der BTS sowie aktive physikalische Kanäle und das zugewiesene Multiframe konfigurieren. Dabei handelt es sich um Konfigurationen auf physikalischer Ebene, die auf dem Hardwaremodell abgebildet werden müssen. Die OML-Routinen wurden implementiert und auf dem physikalischen Modell der virtuellen BTS abgebildet.

Zu Beginn der Arbeit war bereits die Implementierung eines Gerüsts für die virtuelle BTS vorhanden. Das Gerüst umfasste die Einbindung und Konfiguration des Schedulers, die teilweise Implementierung der SAP und OML-Schnittstellen sowie Anpassungen an der VTY-Schnittstelle der BTS. Diese Implementierung wurde als Grundlage übernommen und erweitert. Sie befindet sich unverändert im Branch `laforge/virt-phy` des osmoBTS Github Repositories[8]. Die neue Implementierung, die mit ihrer Gegenseite im osmocomBB Projekt kompatibel ist, befindet sich im Branch `stumpf/virt-phy`. Das Programm kann mit Automake und einem C-Compiler kompiliert und dann mit der ausführbaren Datei `osmo-bts-virtual` gestartet werden. Die Konfiguration erfolgt entweder über die VTY-Konfigurationsschnittstelle, über die im osmoBTS Benutzerhandbuch nachgelesen werden kann, oder direkt über die Konfigurationsdatei im Projektordner. Beispielkonfigurationsdateien sind in `src/osmo-bts-virtual/example-configs` zu finden.

6.3. OsmocomBB mit virtueller Um-Schnittstelle

OsmocomBB ist eine Sammlung von Software rund um eine MS, die in der Regel mit dem Konsolenprogramm Osmocon verwendet wird. Mit Osmocon ist es möglich, die Firmware im flüchtigen Speicher eines der unterstützten Mobiltelefone, wie zum Beispiel dem Motorola C123, auszutauschen. Über eine serielle Schnittstelle kann man das Telefon mit einem Linux Rechner verbinden und von diesem aus steuern, welches der vorkompilierten Firmware Pakete auf das Telefon geladen werden soll. Die einfachste Firmware dürfte `hello_world` sein, die auf dem Display des Mobiltelefons einen Text ausgibt. `rssi_scan` kann die Received Signal Strength Indications (RSSIs), also die Signalstärken der BTS im Umkreis, messen und stellt diese auf dem Display dar. Die umfangreichste Firmware ist `layer1`. Sie implementiert die physikalische Schicht des GSM-Protokollstapels für den Baseband-Prozessor. Für die Steuerung der Funktionen des Baseband-Prozessors und des DSP von außen gibt es eine Kontrollschnittstelle. Diese Schnittstelle wird Layer 1 Control (L1CTL) genannt und ist ein nicht spezifiziertes, osmocomBB-internes Protokoll. Osmocon

[8] `https://github.com/osmocom/osmo-bts.git`

bietet für den Zugriff auf die Schnittstelle ein lokales Linux-Domain-Socket an und leitet den Datenverkehr auf diesem über die serielle Schnittstelle zur `layer1`-Firmware weiter. Der Funktionsumfang des L1CTL-Protokolls entspricht im Großen und Ganzen dem SAP der physikalischen Schicht, die verschiedenen L1CTL-Headertypen den SAP-Primitives. Im Gegensatz zu einer internen SAP-Schnittstelle ist die externe L1CTL-Schnittstelle aber von außen über das Socket zugänglich. Mehrere sogenannte L23-Apps, die die Funktionalität der höheren GSM-Protokollschichten implementieren, nutzen L1CTL für die Steuerung der physikalischen Schicht des verbundenen Mobiltelefons. Die L23-App `ccch_scan` ist ein passiver CCCH und BCCH Sniffer, der empfangene Nachrichten über GSMTAP protokolliert. `cell_log` macht einen vollständigen Messdurchlauf der Empfangsstärken aller ARFCN und gibt das Ergebnis auf der Kommandozeile aus. Besonders interessant ist die L23-App `mobile`, die die Funktionalität eines klassischen Mobiltelefons simuliert und über eine VTY-Schnittstelle gesteuert werden kann. Damit kann der Nutzer zum Beispiel eine Telefonnummer anrufen, eine SMS versenden oder nach BTS in Reichweite suchen.

Die Struktur des Projekts trennt strikt zwischen Host-Software und Target-Software. Im Ordner `src/host` befinden sich alle Anwendungen, die auf dem Linux-Rechner laufen, wie die L23-Apps und Osmocon. In `src/target` wird die Software für das Mobiltelefon gesammelt. Sie muss zuerst cross-kompiliert und dann mit Osmocon geladen werden. Das sind alle Firmware- Implementierungen wie `layer1` oder `rssi_scan`. Da die Anwendung für das virtuelle MS nicht auf dem Mobiltelefon, sondern auf dem Rechner ausgeführt wird, wurde der Ordner `virt_phy` für dessen Entwicklung in `src/host` angelegt. Folgende Grafik zeigt die Schnittstellen der virtuellen physikalischen Schicht (`virthphy`) in osmocomBB im Vergleich mit anderen Target- und Host-Anwendungen.

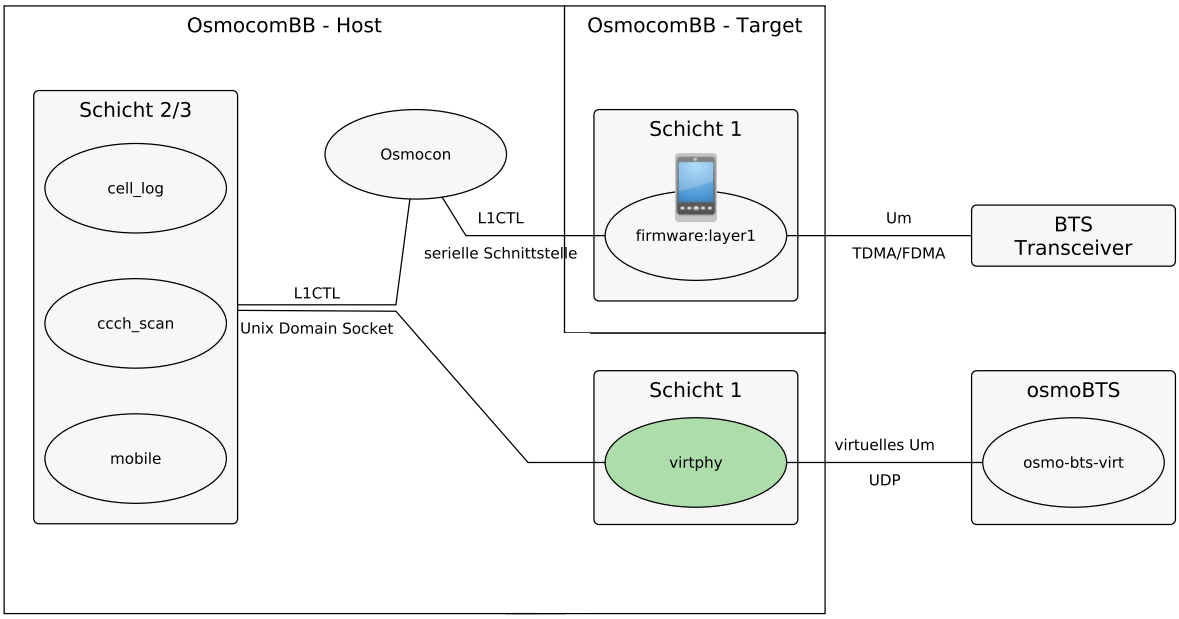


Abbildung 6.3.: Die Integration der virtuellen physikalischen Schicht in osmocomBB, erstellt mit yEd

In Abbildung 6.3 kann man erkennen, dass `virtphy` die strikte Trennung der physikalischen Schicht von den höheren Schichten einhält. `virtphy` implementiert nur die physikalische Schicht der virtuellen MS, die höheren Protokollschichten liegen in den L23-Apps. Die Osmocon Anwendung wurde nicht mit eingebunden, da dessen Hauptfunktionalität, Firmware auf ein Target zu laden, nicht benötigt wird. Neben der Schnittstelle zum virtuellen Um, bietet die virtuelle physikalische Schicht also auch die L1CTL-Schnittstelle über ein Linux-Domain-Socket an. Der Funktionsumfang von `virtphy`, das als eigener Prozess auf dem Rechner läuft, entspricht in etwa der `layer1`-Firmware. Alle Aufgaben die bei der Implementierung der gelöst wurden sind im Folgenden zusammengefasst.

Build-Prozess. Die `virt-phy` Anwendung wurde in den Build-Prozess von osmocomBB eingebunden. Wie in osmoBTS wird Automake verwendet.

Implementierung des L1CTL-Sockets Die Funktionalität zur Erstellung und Konfiguration des L1CTL-Sockets wurde aus Osmocon übernommen und in einer eigenen Datei `l1ctl_sock` gekapselt.

Parsen von GSMTAP. Die GSMTAP-Logik wurde über geteilte Ressourcen aus der virtuellen BTS Implementierung eingebunden.

Implementierung der L1CTL-Schnittstelle. Um das L1CTL-Protokoll implementieren zu können wurde die Kommunikation zwischen der `layer1`-Firmware und verschiedenen L23-Apps analysiert. Von der L1CTL-Schnittstelle gibt es keine offizielle Dokumentation. Die Aufgaben der verschiedenen L1CTL-Routinen wurden durch Debugging und Codeanalyse, sowie teilweise Dokumentation im Code herausgefunden, was deren Implementierung in der virtuellen physikalischen Schicht ermöglichte. Das L1CTL-Protokoll orientiert sich dabei stark am SAP der physikalischen Schicht und definiert verschiedene REQ, IND und CON-Nachrichtentypen (siehe Unterabschnitt 3.3.2). Um Fehler in den L23-Apps zu vermeiden, wurde die Schnittstelle vollständig implementiert und die ausgeführten L1CTL-Routinen an das virtuelle Um angepasst. Unter anderem wurde die Funktionalität der Routinen zur Messungen von Signalstärken, Senden einer Signalisierungsnachricht, dem Wechsel in den dedizierten Modus und der Synchronisation mit einer BTS implementiert. Alle Nachrichten des L1CTL-Protokolls sind mit einer kurzen Beschreibung und dem Status der zugehörigen Routine in der Implementierung der virtuellen physikalischen Schicht, in Abschnitt C.2 hinterlegt.

Routine zur Messung von Signalstärken. Auf dem virtuellen Um gibt es keine unterschiedlichen Signalstärken. Alle Nachrichten auf dem Multicast-Socket werden mit gleicher Qualität empfangen. Da die MS sich jedoch immer mit der BTS mit dem besten Empfang verbindet, ist die Simulation der Signalstärken wichtig. Durch die Konfiguration unterschiedlicher Signalstärken für verschiedene BTS's, kann Einfluss darauf genommen werden, mit welcher BTS sich das MS verbindet. Die Messung von Signalstärken wird über die L1CTL-Nachricht `L1CTL_PM_REQ` von der L23-App, für eine Liste von ARFCNs, gestartet. Die in der Antwort `L1CTL_PM_CON` zurückgegebenen Messwerte der virtuellen physikalischen Schicht wurden so definiert: ARFCNs von denen regelmäßig eine Nachricht auf dem virtuellen Um empfangen wird, erhalten die maximale Signalstärke (> -63 dBm in GSM), alle anderen die minimale Signalstärke (< -110 dBm in GSM). Damit wird gewährleistet, dass die RR-Schicht nur die Verbindung mit verfügbaren ARFCN initiiert. Des Weiteren kann mit Kommandozeilenparametern von `virtphy`, für jede ARFCN ein Signalstärkereduktionswert angegeben werden, um den der zurückgegebene Signalstärkemesswert dieser ARFCN reduziert wird. Damit kann der Anwender unterschiedlichen Empfang für die BTS auf dem virtuellen Um konfigurieren und so beeinflussen, mit welcher BTS sich das MS verbindet.

Routine zur Synchronisation der MS mit einer BTS. Die RR-Schicht in der L23-App schickt die L1CTL-Nachricht `L1CTL_FBSB_REQ`, um sich mit einer ausgewählten ARFCN zu verbinden, sobald ihr die Signalstärkemesswerte aller ARFCNs vorliegen. Im Gegensatz zur realen Funkschnittstelle ist auf dem virtuellen Um keine Frequenzsynchronisation nötig, da auf dem Multicast-Socket die Nachrichten aller BTS empfangen werden. Auf dem virtuellen Um senden die BTS's außerdem keine Nachrichten auf dem SCH, der für die zeitliche Synchronisation mit dem Multiframe verwendet wird. Die Synchronisationsmechanismen wurden deshalb wie folgt an das virtuelle Um angepasst. Die Synchronisation mit einer ARFCN wird durch das Herausfiltern aller Nachrichten von anderen ARFCNs erreicht. Die zeitliche Synchronisation ist auf der realen Funkschnittstelle nötig, um die empfangenen Nachrichten den korrekten logischen und physikalischen Kanälen zuordnen zu können. Auf dem virtuellen Um sind die Informationen, die für die Zuordnung benötigt werden (FN und TDMA-Zeitschlitz), im GSMTAP-Header jeder Nachricht verfügbar. Insofern wird die zeitliche Synchronisation dafür nicht benötigt. Der Scheduler muss für die Planung von Aufgaben für eine FN in der Zukunft aber die aktuelle FN kennen. Um dem Scheduler diese Information zu Verfügung zu stellen, aktualisiert die virtuelle physikalische Schicht, mit jeder eingehenden Nachricht von der synchronisierten BTS, eine lokale FN im Modell der virtuellen MS. Diese wird dem Scheduler zur Verfügung gestellt.

Scheduling. Die Implementierung des Schedulers besteht aus der Planungslogik, die Aufgaben für den richtigen physikalischen und logischen Kanal einplant und der Ausführungslogik der Aufgaben. Der Scheduler wird, sobald die MS mit einer BTS verbunden ist, bei jeder auf dem virtuellen Um empfangenen Nachricht ausgeführt. Der Scheduler ist damit nicht aktiv zeitgesteuert, sondern passiv und abhängig von eingehenden Nachrichten. Da das MS von der BTS jedoch regelmäßig Nachrichten auf dem BCCH und anderen Kanälen erhält, ist diese einfache und passive Lösung ausreichend. Der Scheduler wird mit dieser Lösung nicht für jede FN aufgerufen und führt deshalb der Reihe nach alle Aufgaben aus, die für die aktuelle und kleinere FN geplant wurden. Für den Scheduler wurde bewusst eine einfache Variante gewählt. In den aufgezeichneten Mitschnitten konnte die Lösung die richtige Reihenfolge und FN der gesendeten Nachrichten gewährleisten und war somit ausreichend.

Schnittstelle zum SIM. Für die Ausführung der Sicherheitsalgorithmen benötigt die MS eine Schnittstelle zum SIM. In der `layer1`-Firmware ist diese implementiert

und wird über eine L1CTL-Routine für höhere Schichten zur Verfügung gestellt. Auch das virtuelle MS braucht Zugriff auf Daten und Sicherheitsalgorithmen der SIM-Karte. Dazu kann entweder das SIM simuliert, oder die Schnittstelle zu einem SIM-Kartenlesegerät implementiert werden. In der L23-App `mobile` kann eine `test-sim` Option aktiviert werden, mit der sie eine SIM-Karte simuliert und zur Verfügung stellt. Die L1CTL-Routine für den Zugriff auf die SIM-Karte wurde deshalb nicht implementiert, sondern stattdessen die Aktivierung der `test-sim` Option vorausgesetzt. Damit ein virtuelles MS Zugriff auf alle Netzwerkdienste hat, muss die SIM-Karte einen gültigen Netzteilnehmer identifizieren. Dazu muss die auf der Test-SIM gespeicherte IMSI auch im HLR vorhanden sein. In osmoNITB ist das HLR durch eine SQLite-Datenbank realisiert, in die die Werte der Netzteilnehmer eingetragen werden.

Einige Funktionen der Um-Schnittstelle werden aktuell (Stand Mai 2017) noch nicht von den Implementierungen für das virtuelle MS und BTS unterstützt. Dazu gehören Frequency Hopping, Sprachdatenübertragung, sowie Kodierung und Verschlüsselung des Nachrichtenverkehrs. Für die praktische Durchführung des MitM-Angriffs war die Implementierung dieser Funktionen nicht notwendig. Für die virtuelle physikalische Schicht gibt es zudem keine VTY Konfigurationsschnittstelle. Die Konfiguration erfolgt über Kommandozeilenparameter beim Start des Programms.

6.4. Die Validierung der Manipulation der Setup-Nachricht mit Testdaten

Wegen der fehlenden Kanalkodierung und Verschlüsselung auf dem virtuellen Um wurden das Kommandozeilenprogramm `dummycoder` und das Python Skript `xor_hexstrings` geschrieben. Beide Programme befinden sich im osmoMITM[9] Repository.

Die Kodierungsfunktionalität wurde in `coder.c` implementiert. Für die Kodierung und Dekodierung einer Nachricht auf dem FACCH wurden die Funktionen `facch_encode()` und `facch_decode()` geschrieben, für alle anderen Signalisierungskanäle `xcch_encode()` und `xcch_decode()`. Die Funktionen nehmen als Eingangsdaten das beliebige Ergebnis

[9] `https://github.com/BastusIII/osmo-mitm.git`

eines Zwischenschrittes der Kanalkodierung entgegen und geben die Zwischenergebnisse aller angewendeten Kodierungsfunktionen zurück. Der Grund für die Trennung von FACCH von anderen Signalisierungskanälen liegt in den unterschiedlichen kombinierten Kodierungsverfahren (siehe Abschnitt 3.5). Die Kodierungsverfahren für Firecode und Faltungskodierung werden von der libosmocore Bibliothek zur Verfügung gestellt und wurden aus dieser verwendet. Die Logik für Interleaving und Burstmapping wird nicht von der Bibliothek angeboten, konnte aber aus `osmo-bts-trx` übernommen werden.

Das Programm `dummycoder` liest Hexstrings aus Textdateien ein, kodiert oder dekodiert sie auf Wunsch des Anwenders und speichert das Ergebnis wieder ab. Dafür verwendet es die von `coder.c` bereitgestellten Funktionen und enthält selbst nur die Logik zum Einlesen und Formatieren der Eingangs- und Ergebnisdaten. Das Python Skript `xor_hexstrings` ließt zwei Hexstrings ein und verknüpft diese mit der XOR-Operation. Dadurch lässt sich die Stromverschlüsselung einer GSM-Nachricht simulieren.

Im Ordner `tests` befinden sich zwei Testfälle, in denen die beiden Programme verwendet werden, um eine Beispiel Setup-Nachricht zu kodieren und zu manipulieren. Die Ergebnisse aus `tests/encode_decode_test` sind in die Beispieldaten der Kanalkodierung in Abschnitt 3.5 eingeflossen. Das Skript in `tests/setup_manip_test` wurde verwendet, um die Manipulation der Telefonnummer in einer Setup-Nachricht anhand von Beispieldaten zu validieren. Die Konsolenausgabe des Tests ist in Abschnitt C.3 zu finden.

6.5. Die Validierung des Angriffs mit einem Man-in-the-Middle im virtuellen Um

Für die praktische Nachstellung des in Kapitel 5 ausgearbeiteten Angriffs wurde ein MitM auf dem virtuellen Um implementiert. Der MitM hat Zugriff auf den Uplink und Downlink des virtuellen und eines korrumpierten Um und stellt die Verbindung zwischen beiden her. Alle MS die sich mit dem korrumpierten virtuellen Um verbinden, kommunizieren so über den MitM mit dem Netzwerk und der MitM hat Zugriff auf den kompletten Datenverkehr. Die Implementierung besteht aus einem Rahmen, der die Funktionalität der zwei virtuellen Um-Schnittstellen anbietet, und einer Verarbeitungslogik. Die Verarbeitungslogik wird über zwei als extern deklarierte Callbackfunktionen eingebunden. Der Uplink-Callback wird bei einer eingehenden Nachricht auf dem korrumpierten Uplink aufgerufen, sein Rückgabewert

an das Um weitergeleitet. Der Downlink-Callback wird bei einer eingehenden Nachricht auf dem Downlink aufgerufen, sein Rückgabewert an das korrumpierte Um weitergeleitet. Ein Rückgabewert von NULL bedeutet, dass die Nachricht blockiert wurde. Damit können Nachrichten von der Verarbeitungslogik verändert, verzögert oder abgefangen werden. Abbildung 6.4 zeigt, wie die MitM-Verarbeitungsroutinen im MitM-Gerüst eingebunden sind und die Vermittlung von Nachrichten zwischen virtuellen Um und korrumpiertem virtuellen Um.

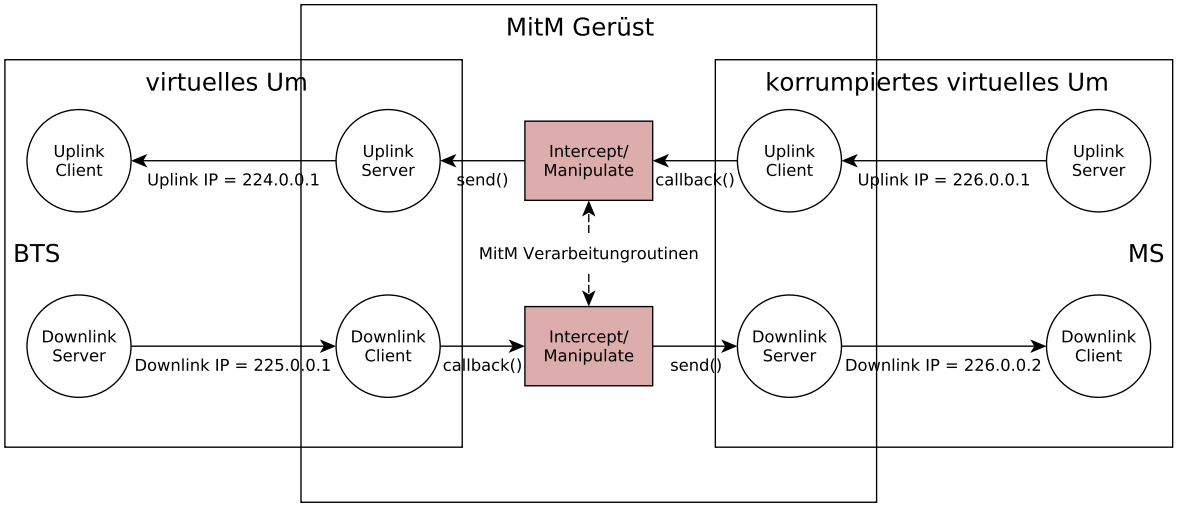


Abbildung 6.4.: Die MitM-Implementierung im virtuellen Um, erstellt mit yEd

Das MitM-Gerüst muss über Kommandozeilenparameter mit den IP-Adressen von Uplink und Downlink des Um und des korrumpierten Um konfiguriert werden, da es die Logik für den Aufbau der virtuellen Um-Schnittstellen enthält. Von den Verarbeitungsroutinen benötigte, zusätzliche Optionen können über einen Optionshandler definiert werden. Eine Verarbeitungsroutine muss neben dem Optionshandler nur die beiden Funktionen implementieren die empfangene Nachrichten analysieren und bearbeiten. Sie kann deshalb leicht ausgetauscht werden. Im Zuge der Masterarbeit wurden drei Verarbeitungslogiken entwickelt. `SimpleForward` leitet alle Nachrichten, ohne sie zu verändern, sobald sie eingehen, sofort weiter. `ImsiCatcher` realisiert einen IMSI-Catcher, der eine GSM Schwachstelle ausnutzt, um die aktuellen TMSIs der Netzteilnehmer ihren IMSIs zuzuordnen und damit ihre Identität offenlegt. `CallSetupManipulation` kombiniert die Funktionalität des IMSI-Catchers mit der im Zuge der Masterarbeit ausgearbeiteten Identifikation und Manipulation der Setup-Nachricht (siehe Kapitel 5), um ein ausgehendes Telefonat umzuleiten. Die einfachste Routine ist in folgendem Codebeispiel zu sehen. Das

C-Struct `msgb` wird innerhalb von Osmocom verwendet, um eine Nachricht mit Nutzdaten und mehreren Headern zu definieren.

```
#include <osmocom/core/msgb.h>

/**
 * Simple Forward hat keine zusätzlichen Optionen.
 */
void handle_suboptions(int argc, char **argv)
{
    return;
}
/**
 * Rückgabe und damit einfache Weiterleitung der originalen Nachricht.
 */
struct msgb* downlink_rcv_cb_handler(struct msgb *msg) {
    return msg;
}
/**
 * Rückgabe und damit einfache Weiterleitung der originalen Nachricht.
 */
struct msgb* uplink_rcv_cb_handler(struct msgb *msg) {
    return msg;
}
```

Codebeispiel 6.2: Die Simple-Forward Verarbeitungsroutine des MitM

Die Implementierung des MitM Frameworks auf dem virtuellen Um mitsamt den Verarbeitungsroutinen ist im osmoMITM[10] Repository hinterlegt.

6.5.1. Die Verarbeitungsroutine des IMSI-Catchers

Die Identitäten auf dem Um werden durch die Verwendung einer temporären ID, der TMSI, verschleiert. Der Angreifer im vorgestellten MitM-Angriff möchte nur die ausgehenden Anrufe von einem bestimmten Netzteilnehmer umleiten. Damit dieser eindeutig vom Angreifer identifiziert werden kann, wird die aktuelle TMSI benötigt, die ihm vom Netzwerk zugewiesen wurde. Da die Zuweisung der TMSI im „Location Updating Accept“ oder „TMSI

[10] `https://github.com/BastusIII/osmo-mitm.git`

Reallocation Command“ verschlüsselt geschieht, kann sie nicht aus dem Datenverkehr auf dem Um ausgelesen werden. Ein sogenannter IMSI-Catcher nutzt Schwachstellen im GSM-Standard aus, um die Identitäten verbundener Netzteilnehmer zu sammeln. Da der Angriff die Kenntnis der Identität des Opfers voraussetzt, wurde das MitM-Gerüst im virtuellen Um um die Logik eines IMSI-Catchers erweitert.

Die Grundlage des IMSI-Catchers ist eine falsche BTS, die auf alle eingehenden Anfragen von verbundenen MS mit einem „Identity Request“ die IMSI des MS abfragt, diese sammelt und einem Angreifer zur Verfügung stellt. Der GSM-Standard spezifiziert, dass eine MS während einer RR-Verbindung eingehende Identity-Requests immer erwarten und beantworten muss [TS-24.008, Kap. 4.3.3.2]. Da der Identity-Request des Angreifers erfolgt, bevor die Verschlüsselung aktiviert wird, muss er sich darum nicht kümmern.

Hat ein Angreifer Zugriff auf den Datenverkehr auf dem Um gibt es zudem mehrere Möglichkeiten, den IMSI-Catcher so zu erweitern, dass er die temporäre Identität (TMSI) von Netzteilnehmern auf dem Um ihrer eindeutigen Identität (IMSI, MSISDN) zuordnen kann. Überwacht der Angreifer mit einem passiven Sniffer den PCH einer BTS, kann er die von einer sogenannten „Silent SMS“ an die MSISDN des Opfers ausgelösten Paging Nachrichten analysieren. In einer Paging Nachricht steht in der Regel die TMSI des Opfers [TS-04.18, Kap. 3.3.2.1.1], die der Angreifer so dessen MSISDN zuordnen kann. Dem Opfer wird der Empfang einer Silent SMS nicht angezeigt, er hat also keine Möglichkeit, den Angriff zu bemerken.

Der in dieser Arbeit implementierte IMSI-Catcher setzt einen aktiven Eingriff in den Datenverkehr auf dem Um voraus. Der von der IMSI-Catcher Verarbeitungsroutine manipulierte Nachrichtenfluss ist in Abbildung 6.5 dargestellt.

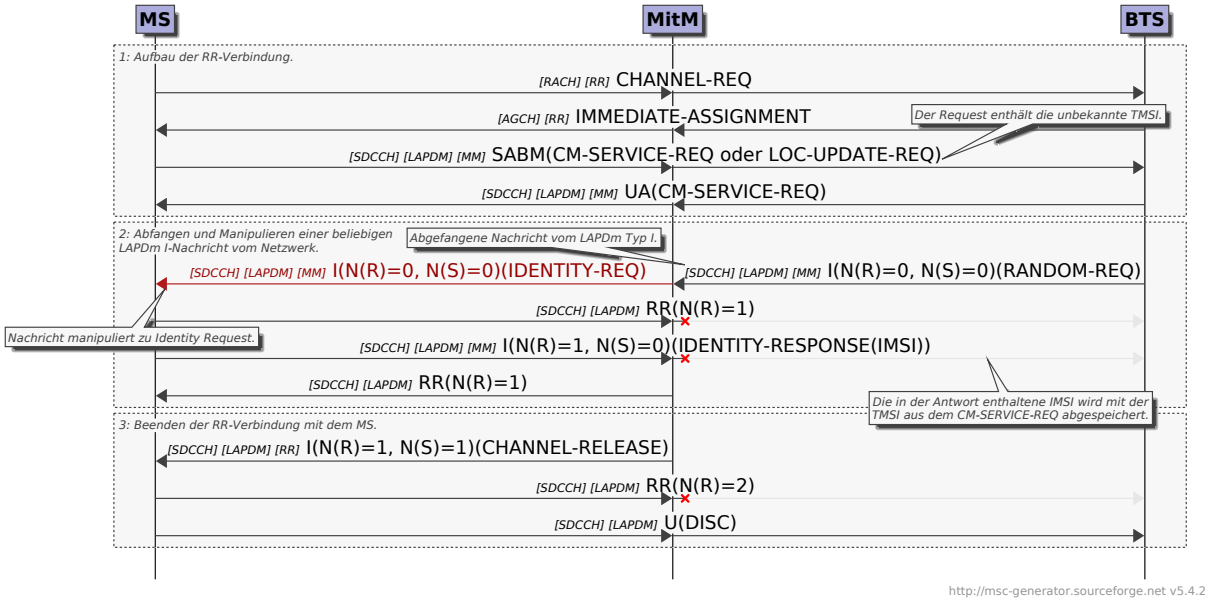


Abbildung 6.5.: Die IMSI-Catcher Verarbeitungsroutine des MitM, verifiziert mit Wireshark-Mitschnitt auf virtuellem Um, siehe Codebeispiel D.3

Die Identität des MS wird in der SABM-Anfrage (siehe Abbildung 6.5, Teil 1) an das Netzwerk entweder als IMSI oder als TMSI angehängt. Ist die angehängte TMSI in einer Anfrage unbekannt, so wird die Nachricht vom Angreifer mit einem Identity-Request beantwortet, der die zugehörige IMSI vom MS abfragt. Die Identity Response mit der IMSI vom MS wird abgefangen und die Zuordnung der IMSI zur TMSI in einer Datenstruktur abgespeichert (siehe Teil 2). Anschließend wird die RR-Verbindung mit einem Channel Release vom MitM beendet (siehe Teil 3). Das MS sendet und empfängt nach dem DISC keine weiteren Nachrichten mehr auf dieser Verbindung. Da das Netzwerk vom Release nichts mitbekommt, horcht es noch eine Weile auf der RR-Verbindung, wird aber, nachdem es eine bestimmte Zeit keine Nachrichten vom MS erhalten hat, von einem Fehler ausgehen und die Verbindung ebenfalls beenden. Da die blockierten Anfragen, wie der Location Update-Request, in der Regel automatisch vom MS generiert werden, bemerkt der Nutzer den Angriff nicht.

Der IMSI-Catcher auf dem virtuellen Um ist wie ein Zustandsautomat aufgebaut. Jeder Nachricht, die als nächstes erwartet wird, wurde ein Zustand zugewiesen. Die beim Zustandsübergang ausgeführten Funktionen definieren, ob und wie die eingegangene Nachricht verändert wird. Mit dem IMSI-Catcher wurden Wireshark Mitschnitte mit erfolgreichen Zuordnungen von TMSI zu IMSI auf dem virtuellen Um erstellt (siehe

Codebeispiel D.3).

6.5.2. Die Verarbeitungsroutine der Setup Manipulation

Die in Kapitel 5 beschriebe Logik für die Manipulation der kodierten und verschlüsselten Setup-Nachricht wurde in dieser Verarbeitungsroutine für den MitM auf dem virtuellen Um implementiert. Das Ziel ist die Umleitung des Anrufs, indem die angerufene Telefonnummer ausgetauscht wird.

Die Funktionalität der Implementierung umfasst die Identifikation des Opfers, wofür die Logik der oben beschriebenen IMSI-Catcher Verarbeitungsroutine eingebunden wurde. Des Weiteren wird die Datenübertragung auf vom identifizierten Opfer ausgehende Anrufe analysiert. Wie die dafür zuständige Setup-Nachricht im Nachrichtenfluss erkannt werden kann, wurde in Abschnitt 5.3 herausgearbeitet. Als Letztes ist die Manipulation der Setup-Nachricht implementiert. Die dafür zuständige Funktion erwartet die LAPDm-Nachricht verschlüsselt und kanalkodiert, wofür die Funktionalität des Coders eingebunden wurde (siehe Abschnitt 6.4). Das Programm wird über Kommandozeilenparameter mit den als bekannt vorausgesetzten Werten initialisiert und erwartet beim Start deshalb die Telefonnummer des angerufenen Opfers (`msisdn-called`), die IMSI des anrufenden Opfers (`imsi-victim`) und das Offset der Telefonnummer in der Call Setup-Nachricht (`msisdn-to-setup-offset`). In folgendem Codebeispiel kann man erkennen, dass die Funktion `manip_setup_msg()`, in der die Manipulation der Setup-Nachricht stattfindet, mit kodierten Daten aufgerufen wird. Die Kodierung übernimmt die `xcch_encode()` Funktion aus dem Coder. Die Daten werden nicht verschlüsselt, was ein zusätzliches XOR mit einem Schlüsselstrom wäre. Der Nachweis, dass der Angriff mit verschlüsselten Daten funktioniert, ist hier nicht mehr nötig. Er wurde in den Tests in Abschnitt 6.4 erbracht.

```
// Auszug aus dem Uplink Callback der Setup Manipulation MitM Verarbeitungsroutine.
// msg enthält die empfangene Originalnachricht.
struct msgb* uplink_rcv_cb_handler(struct msgb *msg)
{
    // Extraktion des GSMTAP Headers aus der Nachricht
    struct gsmtap_hdr *gh = msgb_l1(msg);
    // Puffer für manipulierte Nachricht
    struct msgb *manip_msg;
```

```
    // Puffer für kodierte Originalnachricht. LEN_BURSTMAP_XCCH ist 4*116.
    uint8_t encoded_msg[LEN_BURSTMAP_XCCH / 8];
    // Puffer für kodierte, manipulierte Nachricht
    uint8_t encoded_manip_msg[LEN_BURSTMAP_XCCH / 8];
    (...)
    switch(mitm_state) {
    (...)
    case STATE_IMSI_CATCHER_SABM:
        // Kopieren des GSMTAP Headers (l1h) in den Puffer für die manipulierte Setup
        // Nachricht
        manip_msg->l1h = msgb_put(manip_msg, sizeof(*gh));
        memcpy(manip_msg->l1h, gh, sizeof(*gh));
        (...)
        manip_msg->l1h = msgb_put(manip_msg, sizeof(*gh));
        // Kanalkodierung der Nachricht. Der Kodierungsfunktion aus coder.c wird mit
        // der unkodierten Nachricht vom Typ PLAIN aufgerufen. Das wird Ergebnis im
        // Puffer encoded_msg abgespeichert.
        xcch_encode(PLAIN, msgb_data(msg), encoded_msg, NULL, NULL, NULL);
        // Die Funktion die die Telefonnummer in der Setup-Nachricht austauscht.
        // Die manipulierte Nachricht wird im Puffer encoded_manip_msg abgespeichert
        manip_setup_msg(encoded_manip_msg, encoded_msg, LEN_BURSTMAP_XCCH / 8);
        // Dekodieren der manipulierten Setup-Nachricht. Das Ergebnis wird im msgb
        // Puffer manip_msg an Position des LAPDm Header abgespeichert (l2h).
        xcch_decode(BURSTMAP_XCCH, encoded_manip_msg, NULL, NULL, NULL, manip_msg->l2h
    );
        (...)
        break;
    }
    (...)
    // Der Rückgabewert der Verarbeitungsroutine ist die manipulierte Nachricht,
    // vom MitM Gerüst wird diese also an die BTS weitergeleitet
    return manip_msg;
}
```

Codebeispiel 6.3: Die Callbackfunktion für eingehende Nachrichten auf dem Uplink, Auszug aus der Setup-Manipulation Verarbeitungsroutine des MitM

Mit dem MitM und der Setup Manipulation Verarbeitungslogik konnte der Angriff erfolgreich auf dem virtuellen Um ausgeführt und damit seine Machbarkeit gezeigt werden. Der Wireshark Mitschnitt des Nachrichtenverkehrs auf dem virtuellen Um ist im Anhang in Codebeispiel D.3 zu finden.

7. Das Ergebnis der Arbeit

Das Ergebnis dieser Arbeit ist die Entwicklung und erfolgreiche Durchführung eines praktischen MitM-Angriffs auf GSM. Der Vorteil gegenüber anderen aktuellen Angriffen liegt in seiner Unabhängigkeit vom verwendeten Verschlüsselungsverfahren. Da der Angriff ohne Kenntnis des kryptografischen Schlüssels auskommt, ist das Brechen der Verschlüsselung nicht notwendig. Das ermöglicht seine Ausführung in vollständig mit sicheren Algorithmen wie A5/3 oder A5/4 geschützten Verbindungen.

Der Angriff wurde erfolgreich in einer Testumgebung durchgeführt und seine praktische Machbarkeit dadurch verifiziert. Für die Testumgebung wurden die Osmocom Projekte osmoBTS und osmocomBB, die die BTS und die MS realisieren, modifiziert und um eine virtuelle Um-Schnittstelle erweitert. Für die Durchführung des Angriffs wurde eine MS über eine MitM-Implementierung im virtuellen Um mit der BTS verbunden. Die MS tätigte anschließend einen Anruf an eine bekannte Rufnummer. Die MitM-Implementierung konnte mit der eingebauten IMSI-Catcher Funktionalität das Opfer anhand seiner IMSI erfolgreich identifizieren und den ausgehenden Anruf diesem zuordnen. Über die vor dem Anrufaufbau ausgetauschten, unverschlüsselten Signalisierungsnachrichten zwischen MS und BTS konnte die MitM-Implementierung die für den Anrufaufbau zuständige, verschlüsselte Setup-Nachricht im Nachrichtenfluss erfolgreich identifizieren. Der letzte Schritt des durchgeführten Angriffs bestand in der Manipulation der angerufenen Nummer in der Setup-Nachricht. Ohne Kenntnis des für die Generierung des Schlüsselstroms verwendeten kryptografischen Schlüssels war es möglich, die Telefonnummer in der verschlüsselten und kodierten Setup-Nachricht zu lokalisieren und bekannte Ziffern zu manipulieren. Das Ergebnis des durchgeführten Angriffs war die erfolgreiche Umleitung eines vom Opfer ausgehenden Anrufs an eine vom Angreifer bestimmte Rufnummer.

Da auf der virtuellen Funkschnittstelle GSMTAP-Nachrichten über UDP-Multicast-Sockets kommuniziert werden, konnte der gesamte durchgeführte Angriff von Wireshark aufgezeichnet werden (siehe Codebeispiel D.3).

Die Implementierung der virtuellen Um-Schnittstelle wurde des Weiteren auf der OsmocDevCon2017, der Entwicklerkonferenz des Osmocom Projekts, vorgeführt und positiv aufgenommen. Aktuell liegen die Implementierungen noch in eigenen Branches des osmoBTS und osmocomBB Projekts, sollen aber als nächster Schritt in den Masterbranch integriert werden. Des Weiteren gibt es Pläne, das virtuelle Um als Grundlage für schnelle und hardwareunabhängige Tests im relativ neuen Projekt osmoTester zu integrieren.

8. Verwandte Arbeiten und Angriffe

Neben dem hier vorgestellten Angriff gibt es eine Reihe von weiteren Angriffen auf das GSM-Netzwerk, die Schwachstellen der GSM-Protokolle ausnutzen. Passive Angriffe lauschen nur und greifen nicht in den Datenverkehr ein, bei aktiven Angriffen kann der Angreifer den Datenverkehr manipulieren. In diesem Kapitel wird auf passive und aktive Angriffe eingegangen, die in Bezug zur Arbeit stehen.

8.1. Passive Angriffe

Die Angriffe in diesem Abschnitt beziehen sich auf Schwachstellen der GSM-Protokolle, die es einem passiven Angreifer ermöglichen, an Daten zu gelangen, die nicht für ihn bestimmt sind.

Die erste Kryptoanalyse des A5/1 Algorithmus wurde drei Jahre, nachdem sein Design bekannt wurde, veröffentlicht. Golić [1997] legte darin mehrere Schwachstellen offen und stellte die Idee eines Time-Memory-Trade-Off Angriffs vor, der auf dem Geburtstagsparadoxon beruht. Biryukov u. a. [2001] analysierten den Angriff mit dem Ergebnis, dass er nicht praktikabel war. Er hätte für die Durchführung 15 TB vorberechnete Daten und mehrere Stunden bekannter Gesprächsdaten benötigt. Biryukov u. a. [2001] entwickelten den Angriff in ihrer Arbeit weiter, indem sie mehrere weitere Schwachstellen des A5/1 Algorithmus ausnutzten. Ihr vorgestellter Angriff braucht bis zu 290 GB vorberechneter Daten und hat eine Ausführungszeit von etwa einer Sekunde. Der Nachteil liegt in der mit bis zu zwei Minuten relativ langen Sequenz von Gesprächsdaten, die bekannt sein müssen. Eine weitere Verbesserung des Angriffs auf A5/1 entwickelten Barkan u. a. [2003]. Der Angriff nutzt durch Fehlerkorrekturmechanismen generierte, bekannte Redundanz auf dem TCH aus und kommt deshalb ohne die Voraussetzung von bekannten Gesprächsdaten aus. Die Dauer des Angriffs wurde außerdem auf unter eine Sekunde reduziert und kommt damit

Echtzeitentschlüsselung nahe. Der Nachteil ist die mit mehreren Terabyte große Menge an Daten, die für den Angriff vorberechnet werden müssen. Für die Zeit der Veröffentlichung wäre mit der Berechnung der Daten für die Entschlüsselung eines 5 Minuten Gesprächs ein handelsüblicher Computer mehrere Jahre beschäftigt gewesen, was die praktische Anwendung unrealistisch machte. Auf einer Blackhat Präsentation erwähnte Hulton [2008] die Arbeit an der Berechnung von Rainbow-Tables für einen Time-Memory-Trade-Off Echtzeitangriff auf A5/1. Nohl und Paget [2009] stellte auf der 26C3 den ersten praktisch durchführbaren Angriff auf A5/1 vor. Er schätzte, dass die Berechnungsdauer der Rainbow-Tables von 100.000 Jahren auf einem Prozessor, durch parallele Programmierung mit CUDA[1] und Ausführung auf 80 Graphics Processing Units (GPUs), auf 3 Monate reduziert werden kann. Auf der 27C3 führte er den Angriff mit Munaut praktisch durch [Nohl und Munaut, 2010]. Die Größe der für den Angriff verwendeten Rainbow-Tables konnte auf 2 TB und deren Berechnungsdauer auf einen Monat mit vier GPUs reduziert werden. Im Anschluss an den Vortrag wurden die Tables veröffentlicht[2]. Die Erfolgsrate des Angriffs wird von Nohl auf 99% geschätzt, sofern die Registrierung des Nutzers am Netzwerk mit aufgenommen werden kann, da die bekannten Signalisierungsnachrichten dann maximal sind. Andernfalls sinkt die Erfolgschance auf 50%. Mit dem von Nohl vorgestellten Angriff und den veröffentlichten Rainbow-Tables ist es möglich, einen belauschten und A5/1 geschützten Anruf zu entschlüsseln. Für das Abhören des Telefonats nutzten Nohl und Munaut in ihrer Präsentation zwei Motorola C123 Mobiltelefone (2017 ca. 30 Euro) mit angepasster Firmware aus osmocomBB. Um die Nachrichten auf der Funkschnittstelle zu belauschen, kann auch ein USRP (Ettus Research ca. 1500 Euro) oder ein SDR wie HackRF (ca. 300 Euro) und freie Software[3] genutzt werden.

A5/2 wurde mit der Einführung von GSM für Exportregionen entwickelt und bietet signifikant geringeren Schutz als der zu dieser Zeit als sicher geltende A5/1. Seine kryptografische Schwäche beruht auf einer geringeren verwendeten Schlüssellänge als in A5/1. Goldberg u. a. [1999] veröffentlichten einen Angriff, mit dem das Verschlüsselungsverfahren in Echtzeit gebrochen werden kann. A5/2 wurde einige Jahre, nachdem ein praktischer Angriff von Barkan u. a. [2003] veröffentlicht wurde, von 3GPP offiziell aus der Liste der unterstützten Algorithmen genommen [security.osmocom.org, 2017].

Dunkelman u. a. [2010] publizierten einen praktischen Angriff auf den KASUMI Algorithmus, der in A5/3 für die Generierung des Schlüsselstroms verwendet wird. Wegen der

[1] `http://www.nvidia.de/object/cuda-parallel-computing-de.html`
[2] `https://opensource.srlabs.de/projects/a51-decrypt`
[3] `http://cgit.osmocom.org/gr-osmosdr/`

Art der Verwendung von KASUMI ist der veröffentlichte Sandwich-Angriff aber nicht auf A5/3 anwendbar. KASUMI arbeitet mit einem 128 Bit Schlüssel, da der kryptografische Schlüssel in GSM nur 64 Bit hat, geht dieser also konkateniert (Kc ‖ Kc) in KASUMI ein. Durch die resultierende Entropie des Schlüssels von nur 64 Bit ist A5/3 generell anfällig für Bruteforce Angriffe. Nohl [2014] bezeichnete A5/3 auf der 31C3 als „NSA verwundbar“. Wegen seiner geringen Schlüssellänge kann der Algorithmus Angreifern mit entsprechenden finanziellen Mitteln nicht widerstehen. Laut theintercept.com [2014] arbeitet die National Security Agency (NSA) daran, den A5/3 Algorithmus zu brechen. Auch wenn noch kein praktischer Angriff auf A5/3 veröffentlicht wurde, sollte man sich auf längere Sicht nicht auf dessen Sicherheit verlassen.

Um das Problem der geringen Schlüssellänge zu beheben, wurde A5/4 spezifiziert [TS-55.226]. Dieser nimmt volle 128 Bit für den in KASUMI verwendeten Schlüssel entgegen. Sofern eine 3G USIM-Karte verwendet wird, die einen Schlüssel mit dieser Entropie bereitstellt, ist A5/4 nicht anfällig für Bruteforce Angriffe. 2G SIM-Karte Karten werden von Netzanbietern nicht mehr vertrieben und praktisch nicht mehr verwendet, allerdings gibt es in Deutschland auch noch keinen Mobilfunkanbieter, dessen GSM-Netz A5/4 unterstützt [gsmmap.org, 2016].

Neben den Schwachstellen der Algorithmen gibt es in GSM das Problem, dass Mobiltelefone auch unverschlüsselte Verbindungen akzeptieren, wenn das Netzwerk keine Verschlüsselung bereitstellt. Für passive Angriffe ist die Schwachstelle nicht relevant, da BTS von Netzanbietern Verschlüsselung verlangen, für aktive Angriffe ist sie aber von Bedeutung.

Neben der Verschlüsselung stellt auch Frequency Hopping ein Problem beim Aufzeichnen, Abhören und Manipulieren von Datenverbindungen dar. Der Angreifer muss die verwendete Sprungsequenz der Frequenz kennen und ihr folgen, um den Datenverkehr aufzeichnen zu können. Nohl und Munaut [2010] bemängelten, dass Frequency Hopping in den meisten Fällen nicht genutzt wird. In seiner Präsentation setzte Nohl den Punkt auf seine Wunschliste an die Netzanbieter. Aus Untersuchungen geht hervor, dass in Deutschland zum Stand 2016 Frequency Hopping zwar in der Regel genutzt wird, die verwendeten Sprungsequenzen aber oft vorhersehbar sind. Auch weitere angemerkte Sicherheitsverbesserungen, wie zufälliges Padding, häufiger Wechsel der TMSI und das Aushandeln eines neuen kryptografischen Schlüssels für jeden genutzten Dienst, sind in Deutschland bis 2016 kaum umgesetzt worden [gsmmap.org, 2016].

8.2. Aktive Angriffe

Die Angriffe in diesem Abschnitt beziehen sich auf Schwachstellen im GSM-Protokoll, die es einem aktiven Angreifer ermöglichen, den Nachrichtenverkehr so zu manipulieren, dass er an Informationen kommt, die nicht für ihn bestimmt sind.

Die Grundlage für die meisten vorgestellten aktiven Angriffe ist ein Designfehler im GSM-Authentifizierungsverfahren, der es einem Angreifer ermöglicht, sich Mobiltelefonen gegenüber als BTS auszugeben. Durch das einseitige GSM-AKA kann eine MS die Authentizität der BTS nicht überprüfen. Diese falschen BTS wurden erstmals in Göbel u. a. [1996], als IMSI-Catcher erwähnt. Das ursprünglich von Rohde&Schwarz[4] entwickelte Gerät „GA 090“ wurde von Behörden eingesetzt, um die IMSIs der Netzteilnehmer in Reichweite zu sammeln, wodurch deren Identität und Standort ermittelt werden konnte. Das Sammeln der Identitäten ist durch eine weitere Schwachstelle im GSM-Protokoll möglich. In der Regel wird die IMSI der Teilnehmer auf dem Um verschleiert und durch eine TMSI ersetzt. Die IMSI kann aber jederzeit vom Netzwerk angefordert werden, falls dieses die Zuordnung von IMSI zur TMSI verliert. Da das Netzwerk die Identität und damit den kryptografischen Schlüssel des Teilnehmers nicht kennt, ist der „Identity Request“, mit dem die IMSI abgefragt wird, unverschlüsselt. In GSM ist außerdem spezifiziert, dass eine MS während einer RR-Verbindung einen Identity Request des Netzwerks jederzeit beantworten muss [TS-24.008, Kap. 4.3.3.2]. Ein solcher Identity Request kann vom Angreifer direkt beim von der MS initiierten RR-Verbindungsaufbau geschickt werden, um an die IMSI zu kommen. Durch geringe Modifikationen an der Betriebssoftware des Geräts, konnte der IMSI-Catcher auch Nachrichten zwischen den MS und der echten BTS des Netzanbieters abhören und weiterleiten [Fox, 2002].

Für ihren vorgeschlagenen MitM-Angriff zum Abhören eines verschlüsselten Telefonats setzten Barkan u. a. [2003] eine falsche BTS mit Zugriff auf den Nachrichtenverkehr zwischen MS und Netzwerk voraus. Der Angreifer ist über diese auf der einen Seite mit dem Netzwerk und auf der anderen mit dem Opfer verbunden. Um die falsche BTS als das Opfer beim Netzwerk authentifizieren zu können, wird die Authentifizierungsanfrage des Netzwerks an die MS des Opfers weitergeleitet, die SRES berechnet und dem Angreifer zurückschickt. Mit SRES kann die falsche BTS sich nun ihrerseits beim Netzwerk authentifizieren und sich ihm gegenüber als Opfer ausgeben. Der Angreifer kann die Verschlüsselung zwischen falscher

[4] `https://www.rohde-schwarz.com`

BTS und MS selbst festlegen, unabhängig von der Verschlüsselung, die ihm vom Netzwerk vorgegeben wird. Wählt er einen schwachen Algorithmus wie A5/2, so kann er dessen Schwachstellen ausnutzen, um den von der MS des Opfers generierten kryptografischen Schlüssel Kc zu gewinnen und damit die Daten des Opfers entschlüsseln. Da der selbe kryptografische Schlüssel Kc für alle A5 Algorithmen verwendet wird, können die Daten des Opfers für das Netzwerk mit jeder beliebigen, geforderten Verschlüsselung verschlüsselt werden.

Nohl und Paget [2009] zeigten auf der 26C3, dass es mit günstiger Hardware und freier Software möglich ist, einen IMSI-Catcher umzusetzen. Die praktische Umsetzung des Angriffs führte Paget [2010] ein Jahr später auf der DefCon vor. Er verwendete einen USRP, der über seinen Laptop mit OpenBTS verbunden war, als BTS und Asterisk, sowie Wireshark zum Aufzeichnen und Dekodieren des Datenverkehrs. Er brachte Mobiltelefone dazu, sich mit A5/0, also ohne Verschlüsselung, mit der falschen BTS zu verbinden. Des Weiteren führte er vor, dass durch Stören der UMTS-Frequenzen, MS's trotz Anwesenheit eines 3G Netzes dazu gezwungen werden können, sich auf den GSM-Frequenzen mit der falschen BTS zu verbinden. Auch den MitM-Angriff von Barkan u. a. [2003] konnte er mit seiner falschen BTS praktisch vorführen und die schwachen Algorithmen A5/1 und A5/2 für die Verschlüsselung aushandeln.

Mit dem UMTS-Standard wurde ein gegenseitiges Authentifizierungsverfahren eingeführt, um MitM-Angriffe zu verhindern. Der Datenverkehr auf der Um-Schnittstelle wird in UMTS mit einem Integritätsschlüssel IK geschützt und das Netzwerk authentifiziert sich mit AUTN beim Mobiltelefon. Meyer und Wetzel [2004] stellten jedoch Szenarien vor, in denen ein MitM-Angriff trotz der UMTS-Sicherheitsmechanismen möglich ist. Die Voraussetzung dafür ist, dass das Telefon des Opfers zusätzlich zu UMTS das GSM-Netzwerk unterstützt und eine GSM-BTS in Reichweite ist. Der parallele Betrieb von GSM und UMTS-Infrastruktur ist auch 2017 noch meistens der Fall. Der Angriff macht sich zu Nutze, dass in GSM-Netzwerken auch bei Verwendung des UMTS-AKA kein Integritätsschutz unterstützt wird. Der Angreifer kann so an die Authentifizierungsparameter RAND und AUTN des Netzwerks gelangen und sich der MS gegenüber als dieses ausgeben. Wie in den vorherigen Fällen kann dann eine schwache Verschlüsselung gefordert und ein MitM-Angriff ausgeführt werden.

Neben der Identifikation von Opfern und Angriffen die darauf abzielen verschlüsselte Kommunikation mit dem Netzwerk abzuhören, gibt es eine Reihe von weiteren Anwen-

dungsfällen für eine MitM-BTS auf der Funkschnittstelle. So kann durch binäre SMS mit Over the Air (OTA) Befehlen Schadcode auf der Java Virtual Machine (JVM) von USIM-Karten ausgeführt werden. Dieser kann geheime Daten von Bankanwendungen oder den geheimen Schlüssel des Opfers auslesen und dem Angreifer über SMS zuschicken [Nohl, 2013]. Mit speziellen Nachrichten kann ein Angreifer außerdem auf die Kontrollschnittstelle für die Fernwartung von Mobiltelefonen zugreifen. Damit ist es zum Beispiel möglich, einen neuen Access Point Name (APN) oder Hypertext Transfer Protocol (HTTP)-Proxy einzustellen und so einen permanenten MitM einzurichten [Solnik und Blanchou, 2014].

Die Notwendigkeit von Integritätsschutz für verschlüsselte Daten ist bekannt. Daraus resultierende Sicherheitsprobleme wurden zum Beispiel von Yu u. a. [2004] für den Authentifizierungsdienst Kerberos behandelt. Paterson und Yau [2006] wiesen auf die selben Probleme in der IPSec-Implementierung im Linux Kernel hin, woraufhin Degabriele und Paterson [2007] eine Reihe implementierungsunabhängiger Angriffe auf die IPSec-Spezifikation entwickelten. Von besonderem Interesse für diese Masterarbeit ist der von Bittau u. a. [2006] publizierte Fragmentierungsangriff auf die Stromverschlüsselung von ESP-Paketen in WEP. Der 802.11 Standard spezifiziert, dass ein großes Paket auch fragmentiert mit mehreren kleinen Paketen übertragen werden kann. Bittau und Handley konnten durch den bekannten Klartext des Protokollheaders 8 Byte des Schlüsselstroms extrahieren. Da der 802.11 Standard die Wiederverwendung des Schlüsselstroms für bis zu 16 aufeinanderfolgende Pakete erlaubt, konnten in Kombination mit der Fragmentierung bis zu 64 Byte an Daten eingeschleust werden. Die 8 Byte Fragmente wurden alle mit dem bekannten Teil des Schlüsselstroms verschlüsselt. Das besondere am Angriff ist, dass er nicht darauf abzielt, den kryptografischen Schlüssel herauszufinden, der für die Generierung des Schlüsselstroms verwendet wird. Stattdessen nutzt er einen bekannten Teil des Schlüsselstroms und Schwachstellen des Protokolls aus, um den stromverschlüsselten Datenverkehr zu manipulieren. Der in dieser Arbeit vorgestellte Angriff fällt in die gleiche Kategorie.

Statt einen MitM-Angriff über eine falsche BTS auf der Funkschnittstelle durchzuführen, kann ein Angreifer das Netzwerk auch direkt angreifen. So zeigten Golde u. a. [2012], dass ein Angreifer sich Zugang zu einer Femtozelle verschaffen und diese so manipulieren kann, dass sie für oben erwähnte MitM-Angriffe, wie Verkörperung eines Nutzers und Abhören und Modifizieren des Datenverkehrs, genutzt werden kann. Eine Femtozelle ist eine typischerweise in Privat- oder Firmenumgebungen installierte BTS mit geringer Reichweite, die über einen Digital Subscriber Line (DSL) Anschluss direkt mit dem Netzwerk des Netzanbieters verbunden ist. Ein weiterer Ansatz ist der direkte Angriff auf

das Kernnetzwerk des Netzanbieters. Die Sicherheit dieses SS7-Netzwerks basiert auf dem gegenseitigen Vertrauen aller im Netzwerk agierenden Komponenten, der Datenverkehr wird nicht weiter geschützt. Ein 2014 erschienener Artikel der Washington Post[5] zeigte unter Referenz der Broschüre eines Überwachungsdienstleisters, dass das Szenario eines Angreifers mit Zugriff auf das SS7-Netzwerk Realität ist. Daraufhin beschäftigten sich unter anderem Nohl [2014] und Mourad [2015] mit den Konsequenzen für die Sicherheit des Mobilfunks und stellten eine Vielzahl von MitM-Angriffen vor, die den Zugang zum SS7-Netzwerk voraussetzen und nutzen.

Eine weitere Schwachstelle in GSM ist die Möglichkeit, einen kryptografischen Schlüssel für mehrere Dienste hintereinander benutzen zu können. Ein Angreifer kann so auf Kosten und mit der Identität eines Opfers, Netzwerkdienste in Anspruch nehmen [gsmmap.org, 2016]. Das Vorgeben einer falschen Identität ist ebenfalls durch sogenanntes „Call Spoofing“ möglich, also das Fälschen der Rufnummer des Anrufers. Der Dienst wird von mehreren Webseiten[6] und Smartphone Apps[7] angeboten und ist in Deutschland nach Telekommunikationsgesetz (TKG) §66k Rufnummernübermittlung verboten. Dabei wird der Anruf in der Regel über Voice over IP (VoIP) Dienste geleitet, bei denen die Telefonnummer des Anrufers konfigurierbar ist. Auch Asterisk kann für Call-Spoofing genutzt werden. Die Anwendung erlaubt es dem Angreifer, seine Telefonnummer beliebig zu ändern, bevor ein Anruf an einen VoIP-Dienst weitergeleitet wird.

8.3. Vergleich mit dem neu entwickelten MitM-Angriff

In dieser Masterarbeit wurde der erste MitM-Angriff für GSM entwickelt, der ohne Kenntnis des kryptografischen Schlüssels funktioniert. Der Angriff nutzt, wie der von Bittau u. a. [2006] publizierte Angriff auf WEP, den fehlenden Integritätsschutz aus, um verschlüsselte und bekannte Daten zu manipulieren. In Kombination mit vorhersehbarem Nachrichtenverkehr kann in GSM so die Nachricht für den Anrufaufbau identifiziert und eine bekannte, angerufene Rufnummer geändert werden. Der Angreifer kann den Anruf so an ein Mobiltelefon in seinem Besitz umleiten. Verknüpft der Angreifer den bei seinem Mobiltelefon eingehenden Anruf mit einem Anruf zur ersetzten Nummer, kann er das Gespräch abhören

[5] „For sale: Systems that can secretly track where cellphone users go around the globe“ (Timberg, 2014)
[6] `https://www.spoofcard.com/`
[7] `http://calleridfaker.com/`

und manipulieren – ein MitM-Angriff mit Zugriff auf die unverschlüsselten, zwischen den Opfern kommunizierten Sprachdaten.

Im Gegensatz zu den MitM-Angriffen von Barkan u. a. [2003] und Meyer und Wetzel [2004] beruht der vorgestellte Angriff nicht auf der Verwendung einer schwachen Verschlüsselung wie A5/1 oder A5/2. Da der Angriff keine Kenntnis des kryptografischen Schlüssels voraussetzt, funktioniert er mit allen spezifizierten Verschlüsselungen. Selbst wenn Netzwerk und MS nur absolut sicher verschlüsselte Verbindungen zulassen (z.B. A5/4), kann der MitM-Angriff durchgeführt werden.

Mit der Spezifikation von A5/4 hat 3GPP ein Verschlüsselungsverfahren ohne die in Abschnitt 8.1 gezeigen Schwächen spezifiziert. Der vorgestellte Angriff zeigt auf, dass alle Bemühungen, ein sicheres, auf Stromverschlüsselung basiertes Verschlüsselungsverfahren für GSM zu entwickeln, umsonst sind. Keine Stromverschlüsselung ist in der Lage, den fehlenden Integritätsschutz aufzuwiegen. Solange die Integrität einer Nachricht nicht durch einen sicheren Hashwert geschützt wird, kann ein Angreifer beliebige, ihm bekannte Daten ändern. Der Angriff nutzt diese Schwachstelle aus, um eine bekannte Telefonnummer im Anrufaufbau zu manipulieren. Es wäre jedoch genauso möglich, die oftmals bekannten Inhalte von Signalisierungs- oder anderer Nachrichten zu ändern. Angriffe, die auf solchen oder ähnlichen Manipulationen beruhen, müssen noch untersucht werden.

Mit dem MitM-Angriff ist es mit der Manipulation von einigen wenigen Bits in einer einzigen Nachricht möglich, alle Sicherheits- und Netzwerkmechanismen für den weiteren Verlauf des Gesprächs zu umgehen. Ist der Angriff einmal aufgebaut, kann der Angreifer sich auf viel höherer Ebene mit dem Abhören oder Eingreifen in die Kommunikation beschäftigen - auf Ebene der Audiodaten des Gesprächs. Das Netzwerk und die beteiligten MS's erledigen Verschlüsselung, Kodierung und Frequency-Hopping, sowie den Handover der beiden Opfer und des Angreifers. Selbst ein Handover zu echten Basisstationen, die nicht mehr unter Kontrolle des Angreifers sind, ist möglich, was bedeutet, dass sich Angreifer und Opfer während eines laufenden MitM-Angriffs beliebig bewegen können. Die geringe technische Komplexität des MitM-Angriffs macht ihn als Dienstleistung für Angreifer mit wenig technischem Knowhow interessant. Vorstellbar wäre ein Szenario, in dem Überwachungsdienstleister X anbietet, ausgehende Anrufe von Opfer A an die Nummer von Opfer B über Angreifer Y als MitM zu leiten. Der Dienstleister übernimmt dabei den MitM-Angriff auf dem Um und somit die technisch anspruchsvollere Umleitung des Anrufs und verkauft Angreifer Y den MitM-Angriff auf die Sprachverbindung. Angreifer

Y kann im einfachsten Fall „einfach zuhören“.

Für den MitM-Angriff auf dem Um ergibt sich eine neue Möglichkeit der Umsetzung, verglichen mit der herkömmlichen über eine falsche Basisstation. Wegen der geringen rechnerischen Komplexität der Manipulation der Setup-Nachricht im Vergleich zur Berechnung des kryptografischen Schlüssels könnte auf dem Um ein rein auf physikalischer Ebene, in Echtzeit arbeitender MitM zum Einsatz kommen. Das MitM-Gerät würde in etwa einem GSM-Repeater mit einer beim Empfang und Versand von Nachrichten vorgeschalteten Verarbeitungsroutine entsprechen, die die Bits im Datenstrom flippt. Hardware für GSM-Repeater wird kostengünstig von verschiedenen Onlineshops angeboten [teltarif.de, 2017]. Die Zeit für die Analyse und Manipulation der Daten (sieh Kapitel 5), könnte durch Ausnutzung der Timing Advance Funktionalität gewonnen werden (siehe Abschnitt 3.7). Mit Timing Advance ist der Gewinn von bis zu $233\mu s$ für die Verarbeitung möglich.

Dass die mit UMTS eingeführte, gegenseitige Authentifizierung keinen MitM-Angriff verhindern kann, haben Meyer und Wetzel [2004] gezeigt. Auch der vorgestellte MitM Angriff funktioniert in GSM-Netzwerken mit UMTS-AKA, da dieses in GSM keinen Integritätsschutz bietet und Nachrichten unbemerkt von einem Angreifer manipuliert werden können.

Der vorgestellte MitM-Angriff hat noch zwei Nachteile, die durch Gegenmaßnahmen behoben werden können. Erstens sehen die Opfer die Rufnummer des Angreifers, an den das Telefonat umgeleitet wird. Mit Call-Spoofing kann der Angreifer seine eigene Identität verschleiern und den Opfern die erwartete Identität vorgegeben. Zweitens ergibt sich eine größere Signallaufzeit für die übertragenen Daten, da diese von Opfer 1 zuerst zum Angreifer und dann zu Opfer 2 geschickt werden. Der Angreifer kann den Effekt möglichst gering halten, indem er sich in der Nähe von einem der beiden Opfer aufhält. Die zusätzliche Latenz durch den höheren TA-Wert wegen einem MitM auf der physikalischen Ebene fällt nicht ins Gewicht.

9. Zusammenfassung und Ausblick

Der in dieser Arbeit vorgestellte, praktisch durchführbare MitM-Angriff auf GSM ist der erste seiner Art. Aktuelle existierende MitM-Angriffe, die in den Datenverkehr auf der Funkschnittstelle eingreifen, basieren auf der Verwendung von schwachen Verschlüsselungsverfahren über die der kryptografische Schlüssel gewonnen und die Verschlüsselung gebrochen werden kann. Der entwickelte MitM-Angriff hingegen basiert rein auf der Schwachstelle des fehlenden Integritätsschutzes in GSM und kann unabhängig von verwendeten Verschlüsselungsverfahren durchgeführt werden. Wie schon bei dem von Meyer und Wetzel [2004] veröffentlichten MitM-Angriff schützt auch die Verwendung des gegenseitigen UMTS-Authentifizierungsverfahrens nicht, da es für GSM keinen Integritätsschutz bietet.

Der Angriff kombiniert zwei MitM-Angriffe. Beim ersten wird vorausgesetzt, dass die Kommunikation zwischen dem Mobiltelefon des Opfers und dem Netzwerk über ein vom Angreifer kontrolliertes Gerät auf der Funkschnittstelle läuft. Dieses Gerät kann entweder eine falsche Basisstation sein, oder rein auf physikalischer Ebene arbeiten. Ein MitM-Gerät auf physikalischer Ebene wäre zum Beispiel ein modifizierter GSM-Repeater. Der Angreifer kann aus der unverschlüsselten, vor dem Anrufaufbau notwendigen Kommunikation zwischen einem Netzteilnehmer und dem Netzwerk Rückschlüsse auf die Identität des Anrufers und das exakte Frame, in dem die Setup-Nachricht gesendet wird, schließen. Die Setup-Nachricht enthält Informationen über den ausgehenden Anruf, unter anderem auch die angerufene Telefonnummer. Die Setup-Nachricht ist in der Regel gleich aufgebaut, womit sich die Telefonnummer an einer dem Angreifer bekannten Position befindet. Mit den vorhandenen Informationen kann der Angreifer die Setup-Nachricht eines vom Opfer ausgehenden Anrufs eindeutig identifizieren. Die Nachricht enthält kodierte und verschlüsselte Daten, ist also ohne Kenntnis des kryptografischen Schlüssels nicht ohne weiteres lesbar. Da die verwendeten Kodierungsverfahren bekannt sind, eine Stromverschlüsselung benutzt wird und kein Integritätsschutz vorhanden ist, kann der Angreifer aber bekannte

Originaltextteile beliebig ändern. Der Angriff macht sich das zu Nutze, um die angerufene Nummer zu ändern und den Anruf an ein Mobiltelefon unter Kontrolle des Angreifers weiterzuleiten. Es wird vorausgesetzt, dass die angerufene Telefonnummer oder Teile dieser vom Angreifer in Erfahrung gebracht wurde, zum Beispiel durch Social Engineering. Der Angreifer verknüpft nun den eingehenden Anruf mit einem neuen Anruf bei der ersetzten Nummer und leitet kommunizierte Gesprächsdaten zwischen diesen weiter. Das Ergebnis ist ein MitM-Angriff mit Zugriff auf den unverschlüsselten Gesprächsverkehr zwischen zwei Opfern. Die Daten können unabhängig vom verwendeten Verschlüsselungsverfahren abgehört, aufgenommen und manipuliert werden. Zudem besteht nach dem Anrufaufbau keine Abhängigkeit mehr zum MitM-Gerät auf der Funkschnittstelle. Ein Handover zu einer anderen BTS unterbricht den MitM-Angriff nicht, Opfer und Angreifer können sich frei bewegen.

Der Angriff wurde in dieser Arbeit sowohl theoretisch ausgearbeitet, als auch praktisch in einer virtuellen Testumgebung durchgeführt. Im theoretischen Teil wurde zuerst der für den Anrufaufbau notwendige Nachrichtenverkehr sowie der Inhalt der Setup-Nachricht analysiert. Im Anschluss wurde die Möglichkeit, bekannte Daten in einer GSM kodierten und stromverschlüsselten Nachricht zu manipulieren, mathematisch nachgewiesen. Für die praktische Verifikation des Angriffs wurde freie Software aus dem Osmocom Projekt verwendet. Um hardwareunabhängig testen zu können, wurden die Projekte osmoBTS (die Basisstation) und osmocomBB (das Mobilfunkgerät) um eine virtuelle Luftschnittstelle erweitert. Die physikalische Ebene der Funkschnittstelle wurde dafür durch Multicast-Sockets ersetzt, die GSMTAP gekapselte Nachrichten austauschen. Für den MitM-Angriff wurde eine MitM-Implementierung für das virtuelle Um geschrieben, die einen IMSI-Catcher für die Identifikation des Opfers sowie Funktionalität für die Identifikation und Manipulation der kodierten, verschlüsselten Setup-Nachricht umsetzt. Der Angriff konnte damit erfolgreich durchgeführt und seine praktische Machbarkeit so verifiziert werden.

Was diese Arbeit offenlässt, ist die Durchführung des kompletten vorgestellten MitM-Angriffs. Die Verknüpfung der Audiodaten der beiden Gesprächspartner für den MitM-Angriff beim Mobiltelefon des Angreifers sollte aber keine große Hürde darstellen. Die Machbarkeit des MitM-Angriffs auf der Um-Schnittstelle wurde in einer Testumgebung nachgewiesen. Mit den Erkenntnissen aus dieser Arbeit und basierend auf den Implementierungen wäre der nächste Schritt die praktische Durchführung des Angriffs auf der realen Funkschnittstelle. Interessant wäre dabei insbesondere die Verwendung eines GSM-Repeaters als MitM, da die Eignung und Möglichkeiten dieser Geräte dahingehend noch nicht untersucht wurde.

Auf Basis der durch Timing Advance gewonnenen Zeitspanne für die Echtzeitverarbeitung und Änderung von Daten, sollte er aber möglich sein. Zuletzt wird in dieser Arbeit vorausgesetzt, dass die Länge des Felds für die Bearer-Capability in der Setup-Nachricht bekannt ist. Eine unbekannte Länge würde ein unbekanntes Offset der Telefonnummer bedeuten und deshalb ein Problem für den Angriff darstellen. Das Feld wird für die Übermittlung von unterstützten Sprachkodierungen und weiteren, anrufbezogenen Einstellungen verwendet. Die Länge des Feldes wird im Standard als dynamisch spezifiziert, in Mitschnitten der Testumgebung wurde sie jedoch als konstant beobachtet. Wegen der Funktion des Feldes wird angenommen, dass sein Inhalt vom Telefonmodell eines Herstellers abhängt und so durch die Abfrage der IMEI des Gerätes oder durch Social-Engineering in Erfahrung gebracht werden kann. Diese Annahme muss durch ausreichende Tests mit verschiedenen Mobilfunkgeräten noch nachgewiesen werden.

Das Fazit der Arbeit ist allgemein bekannt [Yu u. a., 2004][Paterson und Yau, 2006][Degabriele und Paterson, 2007][Bittau u. a., 2006]: Verschlüsselung ohne Integritätsschutz gewährleistet keine Vertraulichkeit von Daten. Für GSM bedeutet das im speziellen, dass auf der Um-Schnittstelle übertragene Daten nicht vor Angriffen geschützt sind, auch wenn eigentlich sichere Verschlüsselungsverfahren wie A5/4 verwendet werden. GSM bleibt solange verwundbar, bis Methoden für den Schutz der Datenintegrität spezifiziert und in den Netzwerken verwendet werden. Die Umsetzung eines praktischen, von der Verschlüsselung unabhängigen MitM-Angriffs in dieser Arbeit verdeutlicht diese gravierende Sicherheitsschwachstelle des GSM-Netzwerks.

A. Entwicklung des Mobilfunks

Der Grundstein für GSM wurde bereits 1982 auf der European Conference of Postal and Telecommunications Administrations (CEPT) mit der Gründung der „Groupe Spécial Mobile“ gelegt. Deren Aufgabe war es, einen einheitlichen Standard für die europäischen Mobilfunknetze zu entwickeln. Die Arbeit an der Standardisierung wurde ab 1988 nach und nach von der European Telecommunications Standards Institute (ETSI) übernommen und die „Groupe Spécial Mobile“ ein Komitee der ETSI. 1990 wurden die Spezifikationen des GSM-900-Standards eingefroren und als Basis für den Aufbau der Infrastruktur des Mobilfunknetzes und die Herstellung von Mobiltelefonen verwendet. Ab 1991 begannen die Netzbetreiber, auf Messen für das neue Mobilfunknetz zu werben, der kommerzielle Start des GSM-900 Netzes dauerte aber noch bis 1992. In Deutschland traten als erste Betreiber die Deutsche Telekom (D1 Netz) und Mannesmann Mobilfunk (D2 Netz) auf. 2000 wurden alle GSM-Standards von ETSI nach 3GPP überführt. Aktuell werden sie dort von der GSM EDGE Radio Access Network (GERAN) Arbeitsgruppe weiterentwickelt und gepflegt. Seit der Einführung von GSM wurden die Standards mehrfach überarbeitet, erweitert und an neue Technologien angepasst. Im Folgenden wird die Geschichte der Entwicklung des Mobilfunks nach handy-flatrate-24.de [2017] und 3gpp.org [2017a] kurz aufgeführt.

Ab Ende 50er – Diverse analoge Mobilfunknetze (1G) Die analogen Systeme für Mobilfunk vor GSM waren nicht einheitlich und länderübergreifend standardisiert, teuer und unhandlich. In Deutschland waren das A-, B- und C-Netz der Deutschen Bundespost verbreitet.

1992 – Global System for Mobile Communications (GSM - 2G) Seit 1992 ersetzt GSM als europäischer Standard die vorherigen Mobilfunknetze. Neben Übertragung von Sprachdaten bietet GSM noch die Möglichkeit, Kurzmitteilungen über SMS zu versenden und stellt Roaming Dienste für länderübergreifende Telefonie zur Verfügung. Der Standard wird als 2. Generation (2G) bezeichnet und ist der digitale Nachfolger der veralteten analogen Netze. Mit ca. 700 GSM-Mobilfunknetzen in 200 Ländern ist GSM noch immer eines der verbreitetsten Netze weltweit.

2000 – High Speed Circuit Switched Data (HSCSD) Die Unterscheidung mehrerer Mobiltelefone auf der Luftschnittstelle (siehe Abschnitt 3.3) zur Basisstation funktioniert über TDMA. Eine Verbindung zwischen zwei Mobiltelefonen belegt

dabei einen Kanal, was einem TDMA Timeslot entspricht. Pro Kanal können 14,4 kbit/s übertragen werden. HSCSD ermöglicht es einem Mobiltelefon bis zu 4 Kanäle für die Datenübertragung zu belegen. Dadurch erhöht sich die Auslastung der BTS bei wenigen aktiven Netzteilnehmern und die maximale Transferrate einer Verbindung steigt auf 57,6 kbit/s.

2000 – General Packet Radio Service (GPRS - 2.5G) GSM verwendet Leitungsvermittlung, um eine Verbindung zwischen zwei Netzteilnehmern aufzubauen. Über den reservierten physikalischen Kanal können für die Dauer der Verbindung Daten übertragen werden. Für Telefonie ist das sinnvoll, da während der Dauer eines Gesprächs kontinuierlicher Datenverkehr zwischen den Teilnehmern besteht. GPRS erweitert GSM um paketorientierte Datenübertragung. Statt eine dauerhafte Verbindung zu reservieren, werden zu sendende Daten in Pakete aufgeteilt, die nur bei Bedarf verschickt werden. Damit können mehr Nutzern gleichzeitig Datendienste zur Verfügung gestellt werden und die Kapazität des Netzes wird besser ausgenutzt. GPRS ermöglicht theoretisch Datenraten von 110kBit/s. Um GPRS in das Netzwerk einzubinden, muss das NSS um Service GPRS Support Node (SGSN), Gateway GPRS Support Node (GGSN) und das BSS um eine Packet Control Unit (PCU) erweitert werden (siehe Abschnitt 3.1 - GSM Architektur). Für die BTS müssen zudem neue Kanäle für die Paketdatenübertragung konfiguriert werden. Da die Infrastruktur des GSM-BSS weiter verwendet wird, spricht man von der 2.5ten Generation.

2005/2006 – Enhanced Data Rates for GSM Evolution (EDGE) EDGE führt eine neue Modulationsart ein, welche erhöhte Datenraten für HSCSD und GPRS erlaubt. EDGE erhöht zudem die Stabilität der Datenübertragung. Mit EDGE können bei Belegung von 4 Timeslots bis zu 220kBit/s im Downlink und 110kBit/s im Uplink erreicht werden.

2004 – Universal Mobile Telecommunications System (UMTS - 3G) Der UMTS-Standard wird als 3. Generation bezeichnet, da er nicht länger auf der bestehenden BSS Infrastruktur aufbaut, sondern mit UMTS Terrestrial Radio Access Network (UTRAN) eine neue Infrastruktur benötigt. Das Core Network wird unter dem Namen GERAN größtenteils weiterverwendet. Der Standard schließt Sicherheitslücken in GSM und erhöht Stabilität und Datenrate von Verbindungen. Eine wichtige Änderung war die Überarbeitung des Authentifizierungsverfahrens. Wo das GSM-AKA noch keine Möglichkeit bietet, die Authentizität der BTS zu überprüfen [TS-03.20], wird mit dem UMTS AKA eine gegenseitige Authentifizierung eingeführt [TS-33.102]. Unterstützen AuC des Netzanbieters und Mobiltelefon das neue Authentifizierungsverfahren, so kann es auch in nur 2G-fähigen BSS's verwendet werden – für den GSM-Standard ein deutliches Sicherheitsupgrade. Auf Architektur und Spezifikationen des UMTS-Standards wird in dieser Arbeit nicht näher eingegangen.

2006 – High Speed Packet Access (HSPA - 3.5G) Ein Standard zur weiteren Erhöhung der UMTS Datenraten durch Optimierung der Datenübertragung.

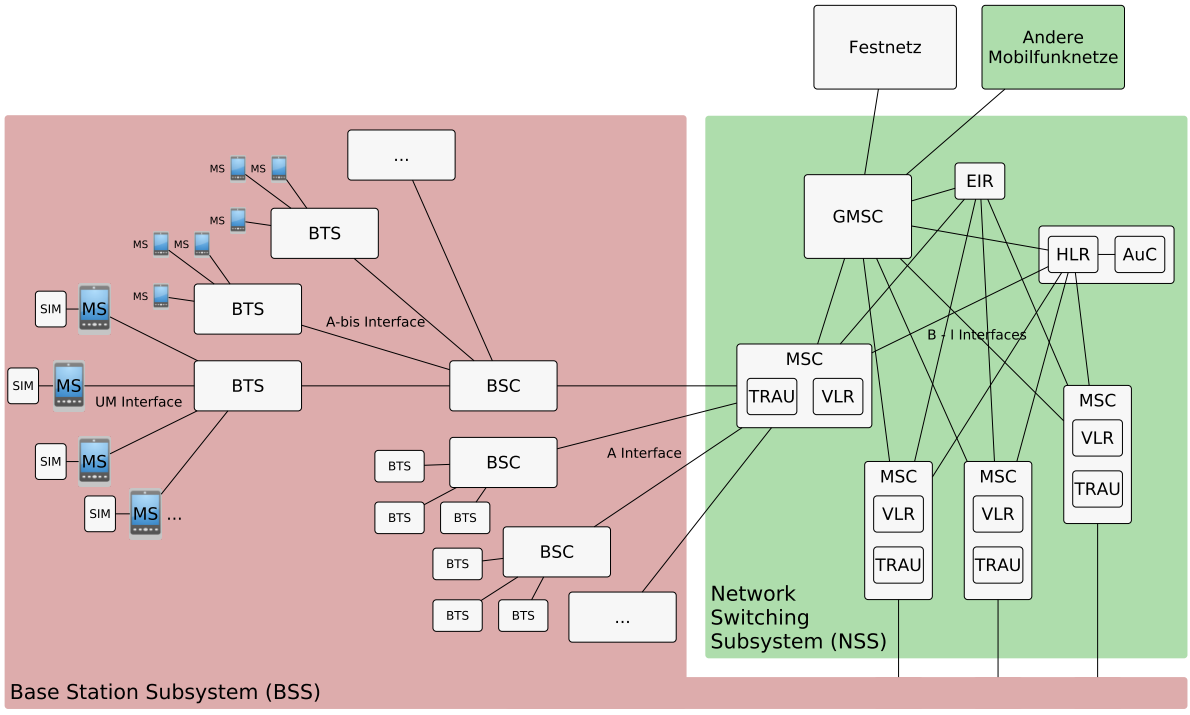
2010 – Long Term Evolution (LTE - 4G) Der aktuellste Mobilfunkstandard mit zukunftssicheren Datenraten bis theoretisch 300Mbit/s, Sicherheit, Stabilität und geringer Latenz. Das Mobilfunknetz der 4. Generation setzt komplett auf paketorientierte Übertragung und ist nicht kompatibel zu UMTS Hardware. Die Netzinfrastruktur muss für flächendeckende LTE Unterstützung also ausgebaut werden. Mit der Voice over LTE (VoLTE) Erweiterung ist auch Sprachübertragung möglich, für die davor auf ältere Standards zurückgegriffen wurde. Vodafone machte 2015 den Anfang, inzwischen unterstützen laut lte-anbieter.info alle deutschen Anbieter VoLTE. Die Weiterentwicklung des Standards, LTE-Advanced, ermöglicht Datenraten bis 1000Mbit/s. Aktuell bieten in Deutschland sowohl Vodafone als auch Telekom eine sehr gute LTE-Abdeckung [ltemap.de, 2017][opensignal.com, 2017]. Auf Architektur und Spezifikationen des LTE-Standards wird in dieser Arbeit nicht näher eingegangen.

2014 – 5G Die Entwicklung eines 5G Standards wurde 2014 von Huawei auf der eigens dafür einberufenen 5G@Europe Summit initiiert. Führende europäische Netzbetreiber forschen seitdem an dem neuen Standard, der mit bis zu 10 GBit/s, niedrigeren Latenzzeiten und Energieverbrauch sowie einer deutlich größeren Kapazität das 4G Netz ablösen soll. Die Standardisierung wurde im Juni 2016 mit Release 15 veröffentlicht und soll im September 2018 abgeschlossen sein [3gpp.org]. Erste Testnetzwerke mit Datenraten von bis zu 3-4 GBit/s wurden auf Messen bereits installiert, für 2020 sind erste Installationen für Endnutzer vorausgesagt [lte-anbieter.info, 2017a].

B. Grundlagen – Anhang

Folgenden Abschnitte umfassen Grundlagen, die für das Verständnis der Arbeit nicht notwendig sind.

B.1. GSM Architektur



TRAU - Transcoding and Rate Adaption Unit Die TRAU transkodiert GSM-kodierte Sprachdaten in das im Netzwerk verwendete ISDN Format (ITU-T A-law). Obwohl sie funktionell dem BSS zugeordnet ist, kann sie entweder zwischen BTS und BSC, oder zwischen BSC und MSC eingebaut werden. Meist wird sie direkt im MSC integriert, da dann weniger TRAU Einheiten benötigt werden und Bandbreite auf dem Abis

und A Interface gespart wird. GSM kodierte Sprachdaten brauchen 13kBit/s, ISDN kodierte 64kBit/s. [Eberspächer u. a., 2008, Kap. 5.2.1]

EIR - Equipment Identity Register Das EIR ist eine optionale Datenbank, die für die Verwaltung von Teilnehmer- und Gerätenummern zuständig ist. Hier werden die IMEIs von Geräten, die als gestohlen gemeldet wurden, auf einer Blacklist geführt. Diesen kann dann ein Zugriff zum Netz verwehrt werden.

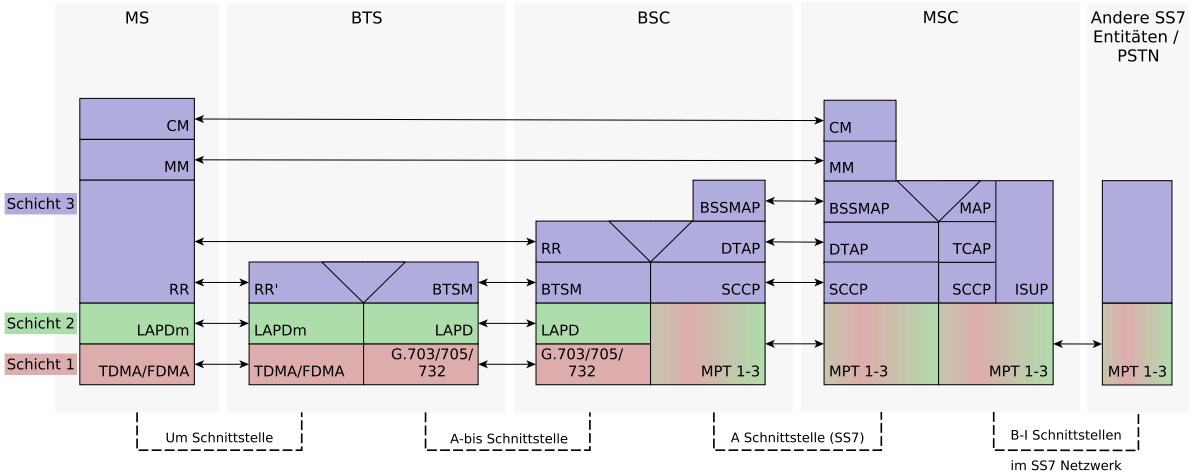
GMSC - Gateway MSC Das GMSC verbindet mehrere MSCs mit anderen Mobilfunknetzen und dem Festnetz. Es ist zuständig für die Vermittlung von Anrufen und Nachrichten zwischen diesen. Für Routing zwischen den Public Land Mobile Networks (PLMNs) hat es innerhalb des SS7 Zugriff auf das HLR und EIR. In der Regel wird ein reguläres MSC zusätzlich an andere PLMNs angebunden und um die Gateway Funktionalität erweitert.

Abis Schnittstelle Die Abis Schnittstelle verbindet BTS und BSC über E-1 Leitungen. Neben der Weiterleitung von Anwenderdaten erlaubt sie über entsprechende Protokolle die Konfiguration des Transceivers sowie Allokation von Frequenzen im BTS.

A Schnittstelle Das A Interface dient dem Datenaustausch zwischen BTS und MSC. Letzteres kann darüber Konfigurationsnachrichten an das BSS schicken und Nutzerdaten weiterleiten.

B-I Schnittstellen Hierbei handelt es sich um Interfaces, die zur Kommunikation zwischen Komponenten des NSS definiert wurden. Die Komponenten sind über ein SS7-Netzwerk miteinander verbunden und kommunizieren über verschiedene Mobile Application Part (MAP) Protokolle.

B.2. GSM Protokolle und Schnittstellen



B.2.1. TDMA/FDMA

GSM verwendet als Phasenmodulation Gaussion filtered Minimum Shift Keying (GMSK), um das Spektrum bei Modulation eines digitalen Signals effektiv auszunutzen [TS-05.04]. Dadurch kann das in GSM-900 zur Verfügung stehende Frequenzband, von 890.0-915.0 MHz für Uplink und 935.0-960.0 MHz für Downlink, in jeweils 124 Trägerfrequenzen mit 200kHz Abstand eingeteilt werden. Über die ARFCNs werden einer BTS Funkkanäle für Uplink und Downlink zugeordnet. [Schnabel, 2003]

In TS-05.02, Kap. 4.3 wird das in GSM verwendete TDMA Verfahren spezifiziert. Ein TDMA Frame (4.615 ms) besteht darin aus 8 Zeitschlitzen mit einer Zeitdauer von 577 μs, sogenannten physikalischen Kanälen. Ein mit jedem TDMA Frame inkrementierter Zähler gibt an, in welchem Frame man sich gerade befindet.

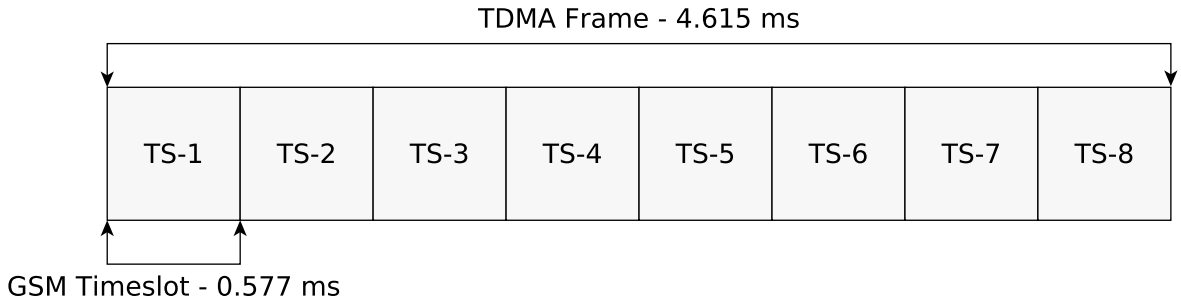


Abbildung B.1.: TDMA-Zeitschlitze in GSM, nach [TS-05.02, Kap. 4.3]

In einem Timeslot kann die Datenmenge von genau einem GSM-Burst übertragen werden. Je nach Inhalt gibt es verschiedene Typen von Bursts, die in TS-05.02, 5.2 nachgelesen werden können. Der Aufbau eines Normal Bursts (NBs) ist in Abbildung 3.3 dargestellt. Neben kodierten Daten enthalten Bursts auch eine „Guard Period“ am Ende, die keine Information mehr enthält, als Puffer zwischen den Timeslots.

Der Uplink wird um 3 Zeitschlitze verzögert mit dem Downlink synchronisiert, damit die MS's genug Zeit haben auf empfangene Daten zu reagieren. Physikalische Kanäle können unterschiedliche Aufgaben wie Broadcasting (BCCH), Signalisierung (CCCH) und Sprachdatenübertragung (TCH) zugewiesen werden. Durch die Konfiguration von bis zu 7 Sprachkanälen pro Trägerfrequenz können je Trägerfrequenz mehrere Sprachverbindungen gleichzeitig übertragen werden. [Schnabel, 2003]

In der Regel wird TS-0 eine Kombination aus BCCH und CCCH, TS-1 CCCH zugeordnet und die restlichen Kanäle für Sprach- oder Datenübertragung verwendet.

B.2.2. Abis Interface

Die BSS-interne Schnittstelle zwischen BTS und BSC ist nicht standardisiert. Das Abis Interface ermöglicht die Steuerung und Konfiguration der BTS über Base Transceiver Station Management (BTSM) und die Weiterleitung transparenter RR, MM und CM Nachrichten.

Die physikalische Schicht besteht meist aus einer kabelgebundenen Verbindung nach TREC-G.703 (oder 705 / 732) mit 2,048 MBit/s (Europa) oder 1,544 MB/s (USA). Die Bandbreite wird über TDMA in 30 Zeitschlitze für Sprache und Daten und zwei Zeitschlitze für Synchronisation und Signalisierung unterteilt.

Für die Datensicherungsschicht wird das ISDN Protokoll LAPD verwendet. Dessen Aufgabe ist der zuverlässige Datentransfer zwischen BTS und BSC auf dem D-Kanal. Der D-Kanal ist in ISDN der Kanal zur Übertragung von Steuerinformationen. Für ISDN ist das Protokoll in TREC-Q.920 und TREC-Q.921 spezifiziert, die GSM Adaption findet man in TS-08.56.

LAPD umfasst folgende allgemeine Funktionen:

- Bereitstellung von einem oder mehreren LAPD-Verbindungen auf dem D-Kanal
- Bereitstellung einer zuverlässigen Verbindung auf dem D-Kanal
- Ablaufkontrolle
- Flußkontrolle
- Fehlererkennung und Korrekturmechanismen
- Benachrichtigung von Layer 3 Einheiten bei fatalen Fehlern

Für Layer 3 werden mehrere LAPD-Verbindungen zur Verfügung gestellt, die über verschiedene SAPIs unterschieden werden. Jede SAPI verwendet dabei ein eigenes Protokoll. Layer 3 auf der A-bis Schnittstelle ist in TS-08.58 beschrieben.

Übersicht über die Protokolle der SAPIs:

SAPI 0, Radio Signaling Link (RSL) wird für Weiterleitung des Datenverkehrs von und an die Luftschnittstelle verwendet, also RR-Signalisierungsnachrichten und die transparenten CC und CM Nachrichten.

SAPI 62, Operation and Maintenance Link (OML) wird für Verwaltung, Konfiguration und Überwachung des BTS und seiner Transceivereinheiten verwendet.

SAPI 63, Layer 2 Management Link (L2ML) ist für die dynamische Verwaltung von Terminal Endpoint Identifiers (TEIs) und der Adressierung der Transceivern zuständig.

B.2.3. A Interface

Das A Interface liegt zwischen BSC und MSC. Der Protokollstack basiert auf dem SS7 Standard, der auch für die Signalisierung im restlichen NSS verwendet wird. Die physikali-

sche Verbindung ist, wie beim Abis Interface, kabelgebunden mit 2,048 oder 1,544 MBit/s. Sie wird über TDMA in 32 Kanäle unterteilt. Davon werden 30 für Datenübertragung und zwei für Signalisierung verwendet. Bei Positionierung der TRAU im BSC TRAU benötigt ein Sprachkanal im ISDN Format 64 kBit/s und füllt damit einen kompletten Slot aus. Es können also bis zu 30 Verbindungen gleichzeitig unterstützt werden. Sitzt die TRAU hingegen im MSC, braucht ein Sprachkanal nur 16 kBit/s. Dadurch sind mehr gleichzeitige Verbindungen möglich.

Schicht 1 und 2 verwenden in SS7 definierte Message Transfer Protocol (MTP), die eine zuverlässige Verbindung zwischen BSC und MSC gewährleisten.

Signaling Connection Control Part (SCCP) ermöglicht auf Schicht 3 eine globale Adressierung von Elementen im SS7-Netzwerk. Mit Base Station Subsystem Management Application Part (BSSMAP) Nachrichten kann das BSS vom MSC konfiguriert und überwacht werden. Des Weiteren stellt SCCP verbindungslose und verbindungsorientierte Nachrichtenflüsse für Radio Resource (RR), CM und MM zur Verfügung.

B.3. A5 Verschlüsselungsverfahren

Von 3GPP aktuell (Stand 2017) spezifizierte, synchrone Verschlüsselungsverfahren auf der Funkschnittstelle.

B.3.1. A5/0

Wird A5/0 ausgehandelt, ist die Verbindung auf der Luftschnittstelle unverschlüsselt. In TS-03.20, 4.8 wird zwar ausdrücklich erwähnt, dass Mobiltelefonen die nur A5/0 unterstützen, abgewiesen werden sollen, umgekehrt gilt das aber nicht. So akzeptieren die meisten Handys eine unverschlüsselte Verbindung, wenn die BTS diese fordert. Dem Benutzer wird das oft nicht signalisiert.

A5/0 wird von echten BTS der Netzbetreiber nicht verwendet, Angreifer mit falschen BTS können diese Schwachstelle aber nutzen.

B.3.2. A5/1

A5/1 ist eine 1987 für GSM entwickelte Stromchiffre und war bis zur Veröffentlichung von A5/3 der sicherste verfügbare Algorithmus.

Die geringe Schlüsselänge von 64 Bit in Kombination mit Schwachstellen gegen Angriffe mit bekanntem Klartext macht den Algorithmus verwundbar [Golić, 1997]. Nohl und Munaut [2010] führte auf der 27C3 vor, dass eine A5/1 verschlüsselte Verbindung mit handelsüblicher und kostengünstiger Hardware in Echtzeit gebrochen werden kann. Da die für den Angriff verwendeten Rainbow Tables veröffentlicht wurden, kann der Angriff seitdem mit geringem eigenen Aufwand von beliebigen Angreifern ausgeführt werden.

A5/1 wird trotz seiner Schwächen auch 2017 noch mit einem Anteil von ca. 50% im GSM-Netz verwendet [gsmmap.org, 2016].

B.3.3. A5/2

A5/2 ist eine Stromchiffre, die 1989 für Exportländer von GSM entwickelt wurde. Mit nur 50 Bit Entropie im kryptografischen Schlüssel ist der Algorithmus deutlich schwächer als A5/1 und wurde von Barkan u. a. [2003] in Echtzeit geknackt.

Um 2007 wurde in den 3GPP Meetings festgelegt, dass A5/2 offiziell nicht mehr von neuen Mobiltelefonen und im GSM-Netz unterstützt werden soll. Alle Netzbetreiber davon zu überzeugen sollte aber noch bis 2008 dauern. [security.osmocom.org, 2017]

B.3.4. A5/3, GEA3

Der A5/3 Algorithmus basiert auf der Blockchiffre KASUMI [TS-35.202] und wurde 2002 in TS-55.216 im Zuge der EDGE/Enhanced GPRS (EGPRS) Entwicklung als Nachfolger von A5/1 spezifiziert. In UMTS wird der Algorithmus standardmäßig unterstützt, in GSM musste er von den Netzbetreibern erst nachgerüstet werden, was den Einzug in GSM-Netzwerken hinauszögerte. In Deutschland wurde A5/3 erst 2013 von der Deutschen Telekom zusätzlich zu A5/1 für GSM eingeführt [telekom.com, 2013].

Abbildung B.2 veranschaulicht das Mapping der Input- und Outputparameter auf die KASUMI-KGCORE-Funktion. Dabei ist Kc der vom A8 aus dem geheimen Schlüssel des Mobilfunkteilnehmers generierte Schlüssel und COUNT die Framenumber der zu verschlüsselnden Nachricht. Die Framenumber kann den maximalen Wert von 2715648 erreichen, wofür die 22 verwendeten Bits ausreichen. A5/3 verwendet KASUMI nur mit 64 Bit Entropie, der von KASUMI verlangte Schlüsselparameter mit 128 Bit wird durch Konkatenation von zwei 64 Bit Schlüsseln erreicht.

Als Ergebnis liefert die Funktion zwei Blöcke BLOCK1 und BLOCK2 mit 114 Bits, die für die Verschlüsselung und Entschlüsselung von je einem Uplink und Downlink

Burst verwendet werden. Verschlüsselt wird über binäre Addition der Datenbits mit dem Schlüsselstrom.

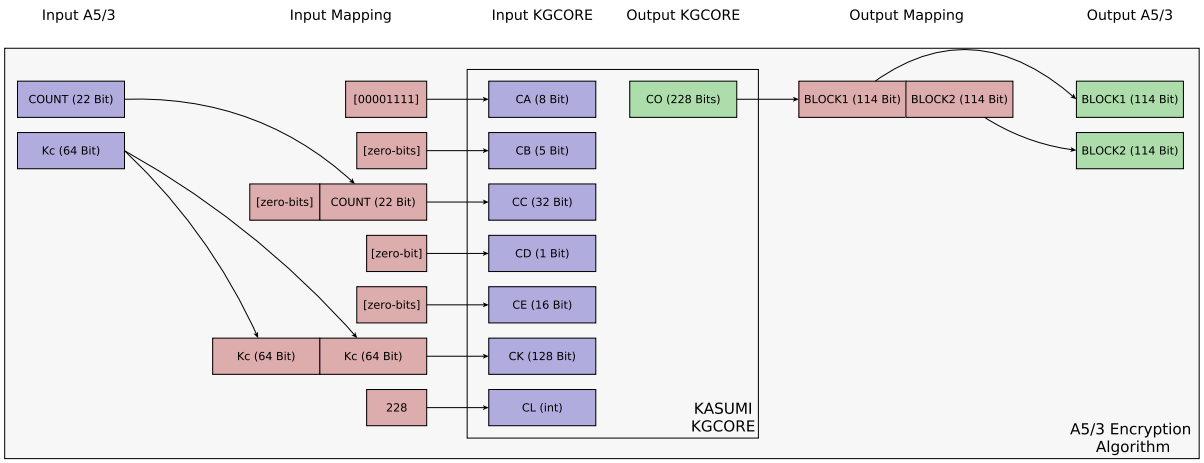


Abbildung B.2.: Das Parameter Mapping von A5/3 auf die Kasumi-KGCORE-Funktion, erstellt mit yEd

Kasumi konnte von Dunkelman u. a. [2010] geknackt werden. Allerdings kann der Angriff nicht auf seine Verwendung im A5/3 Algorithmus angewendet werden. Trotz seiner geringen effektiven Schlüssellänge von 64 Bit gilt er deshalb als relativ sicher. Die Schlüssellänge macht ihn jedoch grundsätzlich verwundbar für Bruteforce Attacken. [Nohl, 2014]

A5/3 ist der sicherste aktuell in GSM-Netzwerken verwendete Algorithmus und deshalb für den vorgestellten Angriff von besonderem Interesse.

B.3.5. A5/4, GEA4

Ein in TS-55.226 spezifiziertes Verschlüsselungsverfahren, das wie A5/3 auf KASUMI basiert, im Gegensatz zu diesem aber die kompletten verfügbaren 128 Bit als Schlüssel verwendet.

Der Algorithmus wird in aktuellen 2G Netzwerken nicht verwendet.

B.3.6. GIA5, GEA5

Ein recht neuer Algorithmus, der 2016 in TS-55.226 spezifiziert wurde.

Der Algorithmus wird in aktuellen 2G Netzwerken nicht verwendet.

B.4. Frequency-Hopping

Frequency-Hopping kann in GSM optional verwendet werden, um für die Übertragung von Daten ein breiteres Frequenzspektrum zu benutzen. Dabei springt die Trägerfrequenz innerhalb eines bestimmten Bereichs. Die Sprungparameter werden in einer bestehenden Verbindung vom BTS vorgegeben und dem Endgerät im SDCCH mitgeteilt.

Damit werden auf der Luftschnittstelle häufige frequenzbedingte Übertragungsprobleme verringert und das Signal-Rausch-Verhältnis verbessert. Frequency Hopping ist - solange eine nicht vorhersehbare Hopping Sequenz verwendet wird - auch eine sicherheitsrelevante Funktion. Die Zuweisung von abgefangenen Signalen zu einer Verbindung wird ohne das Wissen der verwendeten Sprungsequenz erschwert.

GSM definiert in TS-05.02 folgende Parameter für Frequency Hopping:

Mobile Allocation (MA) Eine Liste von erlaubten Sprungfrequenzen für das Mobiltelefon. Es können maximal 63 Frequenzen angegeben werden.

Hopping Sequence Number (HSN) Legt die in der Zelle verwendete Sprungsequenz fest. Es können 64 verschiedene HSNs verwendet werden. HSN = 0 bedeutet zyklisches Springen, die Werte 1 bis 63 geben verschiedene pseudo-zufällige Sequenzen vor.

Mobile Allocation Index Offset (MAIO) Legt die Startfrequenz fest, auf der die MS zu Senden beginnt. Der Wert kann zwischen 0 und der maximal in MA festgelegten Anzahl an unterstützten Frequenzen liegen.

C. Umsetzung – Anhang

In diesem Kapitel werden verschiedene Anhänge mit Bezug zur theoretischen oder praktischen Umsetzung aufgelistet.

C.1. Einrichtung des Testnetzwerks mit virtueller Um-Schnittstelle

In diesem Abschnitt wird die Einrichtung und Konfiguration des Testnetzwerks, in dem der MitM-Angriff durchgeführt wurde, erklärt.

Die Testumgebung wurde in der Masterarbeit auf Linux-Debian Jessie eingerichtet. Folgende Osmocom-Bibliotheken müssen entweder über Github, Autoconf und Automake selbst kompiliert und installiert oder über verfügbare Nigthly-Builds des Paketmanagers installiert werden.

- libosmocore – `git://git.osmocom.org/libosmocore.git`, Branch: master
- libosmo-abis – `git://git.osmocom.org/libosmo-abis.git`, Branch: master
- libosmo-netif – `git://git.osmocom.org/libosmo-netif.git`, Branch: master

Folgende Projekte müssen über Github, Autoconf und Automake selbst kompiliert werden:

- osmoBSC – `git://git.osmocom.org/openbsc.git`, Branch: master
- osmoBTS – `git://git.osmocom.org/osmo-bts`, Branch: stumpf/virt-phy
- osmocomBB – `git://git.osmocom.org/osmocom-bb`, Branch: stumpf/virt-phy
- osmoMITM – `https://github.com/BastusIII/osmo-mitm`, Branch: master

Beim Kompilieren und der Installation obiger Komponenten und ihrer Abhängigkeiten kann man sich an folgenden Anleitungen im Osmocom Wiki orientieren:

- `osmocom.org/projects/cellular-infrastructure/wiki/SDR_OsmoTRX_network_from_scratch`
- `osmocom.org/projects/openbsc/wiki/Building_OpenBSC`
- `osmocom.org/projects/cellular-infrastructure/wiki/Build_from_source`

Sind die Bibliotheken installiert und konnten die Projekte erfolgreich kompiliert werden, geht es an die Konfiguration.

Aus dem osmoBSC Projekt wird die osmoNITB Anwendung benötigt. Für die Verwendung von osmoNITB mit dem virtuellen Um ist keine spezielle Konfiguration notwendig, es kann also die Standardkonfiguration benutzt werden. Es empfiehlt sich aber folgende Parameter anzupassen:

```
network
 auth policy accept-all accept all subscribers
 authorized-regexp .* wildcard needs to be set to really accept all subscribers
 encryption a5 0 don't use encryption, as its currently not supported by virtual Um
 (...)
 bts 0
  type sysmobts declaring the type as sysmobts will work with the virtual bts
  (...)
  trx 0
   (...)
   timeslot 0
    phys_chan_config CCCH+SDCCH4
    hopping enabled 0 hopping should be disabled for all timeslots, not only for this one
    (...)
nitb
 subscriber-create-on-demand create subscribers in hlr automatically
 subscriber-create-on-demand random 1 24
 assign-tmsi activates tmsi assignment and reallocation
```

Aus osmoBTS wird die Anwendung für die virtuelle BTS benötigt. Eine Beispielkonfigurationsdatei ist im Projektordner unter `src/osmo-bts-virtual/example_configs` zu finden. Für die Kommunikation mit dem BSC über die Abis-Schnittstelle muss die `ipa unit-id` mit dem gleichen Wert wie in osmoNITB konfiguriert sein. Informationen dazu finden sich auch in den oben referenzierten Wiki-Links. Für das virtuelle Um müssen folgende Einträge konfiguriert werden, der Rest ist Standard.

```
(...)
phy 0
 virtual-um ms-multicast-group 224.0.0.1 the multicast ip address, the bts will send to
 virtual-um ms-udp-port 4729 4729 will be recognized by Wireshark as GSMTAP
 virtual-um bts-multicast-group 225.0.0.1 the multicast ip address, the bts will receive from
 virtual-um bts-udp-port 4729
(...)
```

Aus osmocomBB wird die Anwendung für die physikalische Schicht des Um `virtphy` und die L23-App `mobile` benötigt. `virtphy` wird über Kommandozeilenparameter erst beim

Aufruf des Programms konfiguriert. Für `mobile` findet sich eine Beispielkonfigurationsdatei im Projektordner in `src/host/virt_phy/example_configs`. Wichtig ist die Aktivierung der `test-sim` Option, da die Implementierung der virtuellen physikalischen Schicht den Zugriff auf die SIM-Application Protocol Data Unit (APDU) nicht über das L1CTL-Socket anbietet.

```
ms 1
 layer2-socket /tmp/osmocom_l2 the path to the L1CTL socket used
 sim test define that the test-sim should be used
(...)
test-sim
 imsi 901700000000403 can be chosen as wished
 ki comp128 12 34 56 78 90 1b cd ef 12 34 56 78 90 ab cd ef can be chosen as wished
 no barred-access
 rplmn 262 42 0x0001 can be used to set the preferred network to avoid the long network search at startup
 hplmn-search everywhere
exit
(...)
```

Der MitM im virtuellen Um aus osmoMITM wird ebenfalls über Kommandozeilenparameter gestartet.

Nach der Konfiguration müssen die Anwendungen in folgender Reihenfolge und mit folgenden Kommandozeilenparametern gestartet werden. Es wird angenommen die Projekte befinden sich alle im Ordner `~/Osmocom` im Home-Verzeichnis des Benutzers und die angepassten Konfigurationsdateien sowie die SQLite-Datenbank für das HLR von osmoNITB in `~/Osmocom/config`.

```
NITB:
~/Osmocom/openbsc/src/osmo-nitb/osmo-nitb
  -c ~/Osmocom/config-files/openbsc-virtual.cfg
  -l ~/Osmocom/config-files/hlr.sqlite3
  -P -C

BTS:
~/Osmocom/osmo-bts/src/osmo-bts-virtual/osmo-bts-virtual
  -c ~/Osmocom/config-files/osmobts-virtual.cfg

MS layer1:
~/Osmocom/osmocom-bb/src/host/virt\_phy/src/virtphy
  --l1ctl-sock /tmp/osmocom\_l2
  --port 4729
  --ul-tx-grp 226.0.0.1
  --dl-rx-grp 226.0.0.2

MS L23-App mobile:
~/Osmocom/osmocom-bb/src/host/layer23/src/mobile/mobile
  -c ~/Osmocom/config-files/osmocom-bb-mobile-virtual-ms1.cfg

MITM:
~/Osmocom/osmo-mitm/src/osmomitmsetupmanip
  --ul-rx-grp 226.0.0.1
```

```
--dl-tx-grp 226.0.0.2
--ul-tx-grp 225.0.0.1
--dl-rx-grp 224.0.0.1
--dump-msgs
--imsi-victim 901700000000403
--msisdn-called 017518181818
--msisdn-attacker 017517171717
--msisdn-to-setup-offset 11
```

Wenn alle Anwendungen korrekt gestartet wurden, kann über Wireshark der Nachrichtenverkehr auf dem virtuellen Um aufgezeichnet werden. Es bietet sich an, die Nachrichten dazu nach `gsmtap` oder `lapdm` zu filtern.

Die Konfigurationen von `nitb`, `bts` und `mobile` können nach dem Start über deren VTY-Schnittstelle eingesehen und angepasst werden. Das VTY von `mobile` wird des Weiteren verwendet, um den Anruf vom Opfer aus zu starten. Ist der MitM wie oben konfiguriert, dann leitet er ausgehende Anrufe der IMSI `901700000000403` an die MSISDN `017518181818` zur MSISDN `017517171717` um. Der Anruf vom Opfer, bzw. `mobile` sollte also an die Telefonnummer `017518181818` erfolgen. Auf das VTY von `mobile` kann, sofern dessen Port nicht verändert wurde, mit telnet zugegriffen werden (`telnet localhost 4247`). Mit dem VTY-Befehl `call 1 017518181818` kann der Anruf gestartet werden.

C.2. L1CTL-Routinen in osmocomBB

L1CTL Routine	Beschreibung	Status
L1CTL_FBSB_REQ/CONF	Cumulated request to sync frequency to a given ARFCN and then start time synchronization.	Implemented
L1CTL_DATA_IND/REQ/CONF	Transmit / receive data on one of the signaling channels FACCH, SDCCH, SACCH.	Implemented
L1CTL_RACH_REQ/CONF	Transmit data on RACH.	Implemented
L1CTL_TRAFFIC_-REQ/CONF/IND	Transmit traffic on a TCH/F or TCH/R.	Implemented
L1CTL_PM_REQ/CONF	Start power measurement on given frequencies.	Implemented
L1CTL_NEIGH_PM_REQ/IND	Start neighbor power measurement.	Not implemented Currently not used as handover is not yet implemented
L1CTL_TCH_MODE_REQ/CONF	Configure TCH mode and audio mode.	Status: Implemented but unused, as we do not support speech recording and transmission yet.
L1CTL_CCCH_MODE_-REQ/CONF	Configure CCCH combined / non-combined mode.	Implemented
L1CTL_RESET_-IND/REQ/CONF	Request or indicate a layer 1 full reset or scheduler reset.	Implemented, used to tell L23 that physical layer has started up e.g.
L1CTL_ECHO_REQ/CONF	Print something out on the display.	Not implemented, but also not needed for the virtual physical layer.
L1CTL_SIM_REQ/CONF	Forward a command to the SIM card.	Not implemented SIM cards are currently not supported. We use the mobile's test-sim option to simulate one.
L1CTL_DM_EST_-REQ/L1CTL_DM_REL_REQ	Handle state chance from idle to dedicated mode and vice-versa.	Implemented
L1CTL_DM_FREQ_REQ	Handle frequency change for a dedicated channel.	Not implemented, as frequency hopping is not yet supported.
L1CTL_PARAM_REQ	Change timing advance value and / or sending power.	Not implemented, but also not needed for the virtual physical layer.
L1CTL_CRYPTO_REQ	Configure the key and algorithm used for cryptographic operations.	Not implemented, as encryption is not yet supported.

Tabelle C.1.: L1CTL-Routinen und deren Implementierungsstand in osmocomBB

C.3. Manipulation von Beispieldaten mit `dummycoder` und `xor_hexstrings.py`

Die Ausgabe wurde von Hand formatiert und farblich gekennzeichnet und wird nicht so von der Testroutine generiert. Die Datenmanipulation und ihre Auswirkung auf die Kodierungsschritte ist rot gekennzeichnet, Kommentare sind blau dargestellt. Die Testroutine kann mit beliebigen Daten ausgeführt werden und liegt im osmoMITM Projekt im Ordner `tests/setup_manip_test`. Das Skript `setup_manip_exec.sh` ruft die Programme `dummycoder` und `xor_hexstrings.py` mehrmals auf, um die Ausgabe zu generieren. Dort müssen die Dateien `data`, `data_manip`, `cipherstream` und `remainder` liegen und mit Daten in Form von Hexstrings befüllt sein. Alle Ergebnisse werden im Ordner `tests/setup_manip_test/test_data` auch als Textdateien abgespeichert.

```
Hinweise:
data:             Originalnachricht als Plaintext
data_manip:       Manipulierte Nachricht als Plaintext
remainder:        Rest des Firecodes in GSM
cipherstream:     Schlüsselstrom
.crc:             Daten nach Anwendung von Firecode
.cc:              Daten nach Anwendung von Convolutional Coding
XORcipherstream:  Stromverschlüsselung

01 20 51 03 45 04 04 60  02 00 81 5e 07 81 10 57
81 81 81 81 15 01 01                           = data

01 20 51 03 45 04 04 60  02 00 81 5e 07 81 10 57
11 81 81 81 15 01 01                           = data_manip

00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
90 00 00 00 00 00 00                           = dataXORdata_manip

00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
90 00 00 00 00 00 00 2f  93 7e 5f 2f 00        = dataXORdata_manip.crc

00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 ff  ff ff ff ff 00        = remainder

00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
90 00 00 00 00 00 00 d0  6c 81 a0 d0 00        = dataXORdata_manip.crcXORremainder

01 20 51 03 45 04 04 60  02 00 81 5e 07 81 10 57
81 81 81 81 15 01 01 f8  b8 1c ef e5 00        = data.crc

01 20 51 03 45 04 04 60  02 00 81 5e 07 81 10 57
11 81 81 81 15 01 01 28  d4 9d 4f 35 00        = dataXORdata_manip.crcXORremainderXORdata.crc

01 20 51 03 45 04 04 60  02 00 81 5e 07 81 10 57
11 81 81 81 15 01 01 28  d4 9d 4f 35 00        = data_manip.crc

00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00        = dataXORdata_manip.crcXORremainderXORdata.crcXORdata_manip.crc

TEST1 - CHECK == 0 OK

00 03 42 3c 37 bc 4f 0e  47 c7 bf 34 f0 34 c9 cc
00 0d 3c 00 d3 c3 78 55  0c 3a 50 c3 4c 4f 37 85
80 4c 9c c3 9c c3 9c c3  4c 78 bf 03 4f 03 42 ef
24 4b 20 6b 4b 19 4d 44  bf                    = [dataXORdata_manip.crcXORremainderXORdata.crc].cc

00 03 42 3c 37 bc 4f 0e  47 c7 bf 34 f0 34 c9 cc
00 0d 3c 00 d3 c3 78 55  0c 3a 50 c3 4c 4f 37 85
50 c3 9c c3 9c c3 9c c3  4c 78 bf 03 4f 03 a6 90
1d 60 c3 a8 da e5 a9 3b  bf                     = data.cc

00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00                     = [dataXORdata_manip.crcXORremainderXORdata.crc].ccXORdata_manip.cc

TEST2 - CHECK == 0 OK

00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

```
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
d0 8f 00 00 00 00 00 00  00 00 00 00 00 00 0d d5
93 81 49 69 3b 56 4e d5  43                      = [dataXORdata_manip].cc

00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 e9 aa
aa aa aa aa aa aa aa aa  43                      = remainder.cc

00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
d0 8f 00 00 00 00 00 00  00 00 00 00 00 00 e4 7f
39 2b e3 c3 91 fc e4 7f  00                      = [dataXORdata_manip].ccXORremainder.cc

00 03 42 3c 37 bc 4f 0e  47 c7 bf 34 f0 34 c9 cc
00 0d 3c 00 d3 c3 78 55  0c 3a 50 c3 4c 4f 37 85
80 4c 9c c3 9c c3 9c c3  4c 78 bf 03 4f 03 42 ef
24 4b 20 6b 4b 19 4d 44  bf                      = [dataXORdata_manip].ccXORremainder.ccXORdata.cc

00 03 42 3c 37 bc 4f 0e  47 c7 bf 34 f0 34 c9 cc
00 0d 3c 00 d3 c3 78 55  0c 3a 50 c3 4c 4f 37 85
80 4c 9c c3 9c c3 9c c3  4c 78 bf 03 4f 03 42 ef
24 4b 20 6b 4b 19 4d 44  bf                      = data_manip.cc

00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00                      = [dataXORdata_manip].ccXORremainder.ccXORdata.ccXORdata_manip.cc

TEST3 - CHECK == 0 OK

12 34 56 78 90 ab cd ef  12 34 56 78 90 ab cd ef
12 34 56 78 90 ab cd ef  12 34 56 78 90 ab cd ef
12 34 56 78 90 ab cd ef  12 34 56 78 90 ab cd ef
12 34 56 78 90 ab cd ef  11                      = cipherstream

12 37 14 44 a7 17 82 e1  55 f3 e9 4c 60 9f 04 23
12 39 6a 78 43 68 b5 ba  1e 0e 06 bb dc e4 fa 6a
42 f7 ca bb 0c 68 51 2c  5e 4c e9 7b df a8 6b 7f
0f 54 95 d0 4a 4e 64 d4  ae                      = data.ccXORcipherstream

12 37 14 44 a7 17 82 e1  55 f3 e9 4c 60 9f 04 23
12 39 6a 78 43 68 b5 ba  1e 0e 06 bb dc e4 fa 6a
92 78 ca bb 0c 68 51 2c  5e 4c e9 7b df a8 8f 00
36 7f 76 13 db b2 80 ab  ae                      = data_manip.ccXORcipherstream

00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
d0 8f 00 00 00 00 00 00  00 00 00 00 00 00 e4 7f
39 2b e3 c3 91 fc e4 7f  00                      = [dataXORdata_manip].ccXORremainder.cc

12 37 14 44 a7 17 82 e1  55 f3 e9 4c 60 9f 04 23
12 39 6a 78 43 68 b5 ba  1e 0e 06 bb dc e4 fa 6a
92 78 ca bb 0c 68 51 2c  5e 4c e9 7b df a8 8f 00
36 7f 76 13 db b2 80 ab  ae                      = [dataXORdata_manip].ccXORremainder.ccXORdata.ccXORcipherstream

00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00                      = [dataXORdata_manip].ccXORremainder.ccXORdata.ccXORcipherstreamXOR
                                                   data_manip.ccXORcipherstream

TEST4 - CHECK == 0 OK
```

Codebeispiel C.1: Die Kommandozeilenausgabe von `setup_manip_test` aus osmoMITM

D. Relevante GSM-Abläufe

Im Folgenden werden verschiedene Abläufe in GSM anhand von Nachrichtenflussdiagrammen oder Wireshark-Mitschnitten aufgelistet.

D.1. Radio Resource Connection Establishment

Neben den BCCH und CCCH gibt es in GSM noch die dedizierten logischen Kanäle, bei denen eine Verbindung zwischen je einem MS und einem BTS aufgebaut wird. Der interne Nachrichtenaustausch zwischen den Protokollschichten läuft über SAPs.

Um bidirektional über einen solchen Kanal kommunizieren zu können, ist der Aufbau einer LAPDm und darauf aufbauend einer RR Verbindung notwendig. Die Signalisierung, die zur Einrichtung und Auflösung einer RR-Verbindung nötig ist, wird in Abbildung D.1 dargestellt.

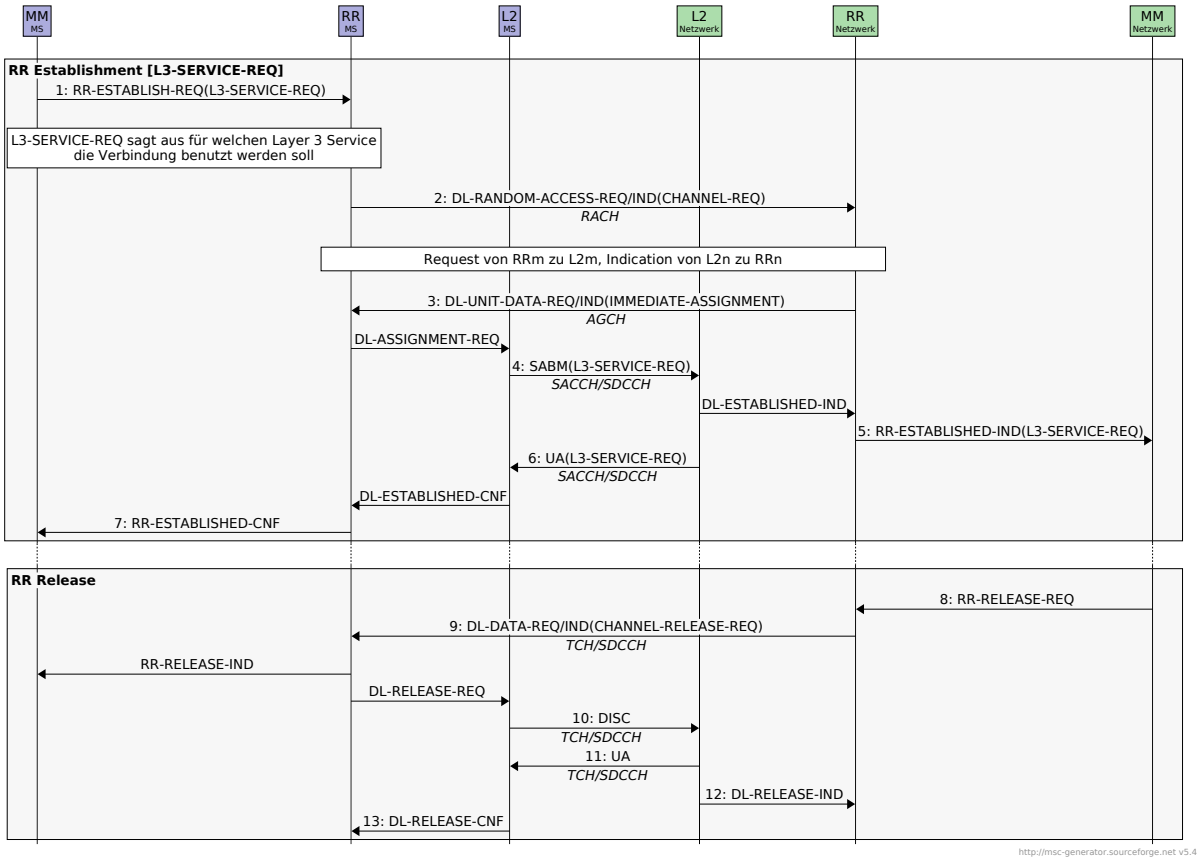


Abbildung D.1.: Nachrichtenfluss von RR-Verbindungsaufbau und Release

1 Der Verbindungsaufbau wird durch ein `RR-ESTABLISH-REQ` Primitive vom MM_{MS} an das SAP des RR_{MS} in Auftrag gegeben. Entweder initiativ vom MS, oder als Reaktion einer eingehenden Paging Nachricht auf dem PCH.

2 RR_{MS} lässt daraufhin von der physikalischen Schicht auf dem RACH eine `CHANNEL-REQUEST`-Nachricht an $RR_{Netzwerk}$ schicken. Wir sehen in Abbildung 3.5, dass diese einen SAP direkt für RR zur Verfügung stellt, die Datensicherungsschicht LAPDm wird dabei übergangen.

3 $RR_{Netzwerk}$ reserviert entsprechende Ressourcen auf einem der dedizierten Kanäle. Das ist bei Early und Late-Assignment ein SDCCH und bei Very-Early-Assignment bereits der TCH. Über den AGCH werden RR_{MS} in einer `IMMEDIATE-ASSIGNMENT`-Nachricht anschließend Informationen zum reservierten Kanal mitgeteilt. Schicht 2 wird wieder übergangen.

4 Nach Erhalt des Immediate-Assignments wechselt das MS in den „Dedicated Mode“ und empfängt nur noch Nachrichten auf dem zugewiesenen dedizierten Kanal. BCCH,

CCCH werden ignoriert. Eine zuverlässige Verbindung (Unterabschnitt 3.3.3) wird von $L2_{MS}$ über die LAPDm-Nachricht SABM im unacknowledged Mode initiiert [TS-04.06, 3.8.2].

5 SABM trägt dabei den kompletten von MM_{MS} in Auftrag gegebenen Service Request (`L3-SERVICE-REQUEST`) mit sich, was „piggybacking“ genannt wird. Nachdem $L2_{Netzwerk}$ über den SAP mit einer `DL-ESTABLISHED-IND` $RR_{Netzwerk}$ über die Einrichtung der zuverlässigen Verbindung informiert hat, schickt dieses den Service Request weiter an $MM_{Netzwerk}$.

6 Mit der LAPDm-Nachricht UA wird $L2_{MS}$ angewiesen, nun in den Acknowledged Information Transfer Mode zu wechseln. Ab dann werden durchnummerierte I-Frames geschickt und die Verbindung ist durch die Retransmission Prozeduren, Acknowledgement (ACK) und REJ-Nachrichten von Schicht 2 vor Paketverlust geschützt.

7 Durch entsprechende SAP Confirm Primitives werden RR_{MS} und MM_{MS} über die aufgebaute zuverlässige Verbindung informiert. Schicht 3 kann nun über LAPDm auf diese zugreifen und Daten übertragen. Im Anschluss kann die Signalisierung für Authentifizierung und Verschlüsselung ablaufen, wofür die zuverlässige Verbindung benötigt wird. [TS-04.18, 3.3] [TS-24.007, Figure A.1 ff.]

8 In der Abbildung initiiert $MM_{Netzwerk}$ das Auflösen der Verbindung mit einem `RR-RELEASE-REQ`, was aber auch vom MS ausgehen kann.

9 Der `CHANNEL-RELEASE-REQUEST` wird an RR_{MS} weitergeleitet.

10 RR_{MS} reagiert indem es MM_{MS} über die beendete Verbindung informiert und $L2_{MS}$ befiehlt, diese aufzulösen. $L2_{MS}$ kommt dem nach und informiert über LAPDm `DISC` $L2_{Netzwerk}$ über den Wechsel in den Unacknowledged Information Transfer Mode, der mit einem UA Frame bestätigt wird.

11 Erhält RR_{MS} die Bestätigung über das Beenden der zuverlässigen Verbindung, wechselt das MS vom „Dedicated“ wieder in den „Idle Mode“. Es werden also wieder Nachrichten auf BCCH und CCCH empfangen und bearbeitet. [TS-04.18, 3.4.13] [TS-24.007, Figure A.1 ff.]

D.2. Wireshark Mitschnitte

Im Folgenden sind Mitschnitte des Nachrichtenverkehrs auf dem Um aufgelistet. Mitschnitte Codebeispiel D.2 und Codebeispiel D.1 wurden von osmocomBB mit aktivem GSMTAP

Logging erstellt. Die Destination-IP-Adresse ist deshalb immer das lokale Netzwerkinterface. Codebeispiel D.3 zeigt den aufgezeichneten MitM-Angriff auf dem virtuellen Um. Dabei sind die MS unter der IP-Adresse 226.0.0.1 zu erreichen, der MitM unter 226.0.0.1 und die BTS. Da sowohl die Verbindung von MS zu MitM, als auch von MitM zu BTS aufgezeichnet wird, scheinen manche Nachrichten doppelt zu sein. In Codebeispiel D.3 wurden die Details von interessanten Nachrichten aufgeklappt.

Der in Wireshark angewendete Filter für den detaillierten Nachrichtenverlauf mit LAPDm-Kontrollnachrichten ist:

```
   !(gsmtap.chan_type == 137)
&& !(gsmtap.chan_type == 136)
&& !(gsm_a.dtap.msg_rr_type == 0x15)
&& lapdm
```

Für den detaillierten Nachrichtenverlauf mit LAPDm-Kontrollnachrichten ist der Filter:

```
   !(lapdm.length_field == 0x01)
&& !(gsmtap.chan_type == 137)
&& !(gsmtap.chan_type == 136)
&& !(gsm_a.dtap.msg_rr_type == 0x15)
&& lapdm
```

Kommentare in den Mitschnitten werden in blau angezeigt und sind nicht Teil der Nachrichten. Manipulierte Nachrichten in Codebeispiel D.3 sind rot gekennzeichnet.

```
No.     Destination           Protocol
    293 127.0.0.1             LAPDm    U P, func=SABM(DTAP) (MM) Location Updating Request
    298 127.0.0.1             LAPDm    U F, func=UA(DTAP) (MM) Location Updating Request
    299 127.0.0.1             LAPDm    I, N(R)=0, N(S)=0(DTAP) (MM) Identity Request
    301 127.0.0.1             LAPDm    I, N(R)=1, N(S)=0(DTAP) (MM) Identity Response
    303 127.0.0.1             LAPDm    I, N(R)=0, N(S)=1(DTAP) (MM) Identity Request
    307 127.0.0.1             LAPDm    I, N(R)=2, N(S)=1(DTAP) (MM) Identity Response
    309 127.0.0.1             LAPDm    I P, N(R)=1, N(S)=1(DTAP) (MM) Identity Request
    313 127.0.0.1             LAPDm    I, N(R)=2, N(S)=2(DTAP) (MM) Authentication Request
    323 127.0.0.1             LAPDm    I, N(R)=3, N(S)=2(DTAP) (MM) Authentication Response
    329 127.0.0.1             LAPDm    I, N(R)=3, N(S)=3(DTAP) (RR) Ciphering Mode Command
    331 127.0.0.1             LAPDm    I, N(R)=4, N(S)=3(DTAP) (RR) Ciphering Mode Complete
    333 127.0.0.1             LAPDm    I, N(R)=4, N(S)=4(DTAP) (MM) Identity Request
    335 127.0.0.1             LAPDm    I, N(R)=5, N(S)=4(DTAP) (MM) Identity Response
    339 127.0.0.1             LAPDm    I, N(R)=5, N(S)=5(DTAP) (MM) Location Updating Accept
    341 127.0.0.1             LAPDm    I, N(R)=6, N(S)=5(DTAP) (MM) TMSI Reallocation Complete
    348 127.0.0.1             LAPDm    I, N(R)=5, N(S)=6 (Fragment)
    353 127.0.0.1             LAPDm    I P, N(R)=6, N(S)=6 (Fragment)
    355 127.0.0.1             LAPDm    I, N(R)=6, N(S)=7(DTAP) (MM) MM Information
    357 127.0.0.1             LAPDm    I, N(R)=6, N(S)=0(DTAP) (RR) Channel Release
   2987 127.0.0.1             LAPDm    U P, func=SABM(DTAP) (MM) CM Service Request
   2992 127.0.0.1             LAPDm    U F, func=UA(DTAP) (MM) CM Service Request
   2994 127.0.0.1             LAPDm    I, N(R)=0, N(S)=0(DTAP) (MM) Identity Request
   2996 127.0.0.1             LAPDm    I, N(R)=1, N(S)=0(DTAP) (MM) Identity Response
   2997 127.0.0.1             LAPDm    I, N(R)=0, N(S)=1(DTAP) (MM) Authentication Request
   3007 127.0.0.1             LAPDm    I, N(R)=2, N(S)=1(DTAP) (MM) Authentication Response
   3008 127.0.0.1             LAPDm    I P, N(R)=1, N(S)=1(DTAP) (MM) Authentication Request
   3012 127.0.0.1             LAPDm    I, N(R)=2, N(S)=2(DTAP) (RR) Ciphering Mode Command
   3014 127.0.0.1             LAPDm    I, N(R)=3, N(S)=2(DTAP) (RR) Ciphering Mode Complete
```

```
3023 127.0.0.1          LAPDm    I, N(R)=0, N(S)=0 (Fragment)
3025 127.0.0.1          LAPDm    I, N(R)=0, N(S)=1 (Fragment)
3029 127.0.0.1          GSM SMS  I, N(R)=0, N(S)=2(DTAP) (SMS) CP-DATA (RP) RP-DATA (MS to Network)
3030 127.0.0.1          LAPDm    I, N(R)=3, N(S)=0(DTAP) (SMS) CP-ACK
3033 127.0.0.1          GSM SMS  I, N(R)=3, N(S)=1(DTAP) (SMS) CP-DATA (RP) RP-ACK (Network to MS)
3037 127.0.0.1          LAPDm    I, N(R)=2, N(S)=3(DTAP) (SMS) CP-ACK
3040 127.0.0.1          LAPDm    I, N(R)=3, N(S)=3(DTAP) (RR) Channel Release
4630 127.0.0.1          LAPDm    U P, func=SABM(DTAP) (MM) CM Service Request
4635 127.0.0.1          LAPDm    U F, func=UA(DTAP) (MM) CM Service Request
4637 127.0.0.1          LAPDm    I, N(R)=0, N(S)=0(DTAP) (MM) Identity Request
4639 127.0.0.1          LAPDm    I, N(R)=1, N(S)=0(DTAP) (MM) Identity Response
4640 127.0.0.1          LAPDm    I, N(R)=0, N(S)=1(DTAP) (RR) Ciphering Mode Command
4645 127.0.0.1          LAPDm    I, N(R)=2, N(S)=1(DTAP) (RR) Ciphering Mode Complete
4646 127.0.0.1          LAPDm    I P, N(R)=1, N(S)=1(DTAP) (RR) Ciphering Mode Command
4650 127.0.0.1          LAPDm    I, N(R)=2, N(S)=2(DTAP) (CC) Setup
4654 127.0.0.1          LAPDm    I, N(R)=3, N(S)=2(DTAP) (CC) Call Proceeding
4658 127.0.0.1          LAPDm/GSM MAP I, N(R)=3, N(S)=3(DTAP) (CC) Facility (GSM MAP) invoke notifySS
4661 127.0.0.1          LAPDm    I, N(R)=3, N(S)=4 (Fragment)
4663 127.0.0.1          LAPDm    I, N(R)=3, N(S)=5(DTAP) (RR) Assignment Command
4668 127.0.0.1          LAPDm    I, N(R)=0, N(S)=0(DTAP) (RR) Assignment Complete
4670 127.0.0.1          LAPDm    I, N(R)=1, N(S)=0(DTAP) (CC) Progress
4693 127.0.0.1          LAPDm    I, N(R)=1, N(S)=1(DTAP) (CC) Progress
4695 127.0.0.1          LAPDm    I, N(R)=1, N(S)=2(DTAP) (CC) Alerting
4732 127.0.0.1          LAPDm    I, N(R)=1, N(S)=3(DTAP) (CC) Connect
4736 127.0.0.1          LAPDm    I, N(R)=4, N(S)=1(DTAP) (CC) Connect Acknowledge
4777 127.0.0.1          LAPDm    I, N(R)=2, N(S)=4(DTAP) (CC) Disconnect
4781 127.0.0.1          LAPDm    I, N(R)=5, N(S)=2(DTAP) (CC) Release
4783 127.0.0.1          LAPDm    I, N(R)=3, N(S)=5(DTAP) (CC) Release Complete
```

Codebeispiel D.1: Aufgezeichneter Nachrichtenverkehr mit E-plus BTS, kurz (ohne LAPDm-Kontrollnachrichten), generiert mit Filtern aus Wireshark-Mitschnitt

```
No.     Destination          Protocol
    293 127.0.0.1            LAPDm    U P, func=SABM(DTAP) (MM) Location Updating Request this RR-connection is for the LU
    298 127.0.0.1            LAPDm    U F, func=UA(DTAP) (MM) Location Updating Request
    299 127.0.0.1            LAPDm    I, N(R)=0, N(S)=0(DTAP) (MM) Identity Request
    300 127.0.0.1            LAPDm    S, func=RR, N(R)=1
    301 127.0.0.1            LAPDm    I, N(R)=1, N(S)=0(DTAP) (MM) Identity Response
    303 127.0.0.1            LAPDm    I, N(R)=0, N(S)=1(DTAP) (MM) Identity Request
    304 127.0.0.1            LAPDm    S, func=RR, N(R)=2
    306 127.0.0.1            LAPDm    S, func=RR, N(R)=1
    307 127.0.0.1            LAPDm    I, N(R)=2, N(S)=1(DTAP) (MM) Identity Response
    309 127.0.0.1            LAPDm    I P, N(R)=1, N(S)=1(DTAP) (MM) Identity Request
    310 127.0.0.1            LAPDm    S F, func=REJ, N(R)=2 identity response rejected by lapdm-> needs to be retransmitted
    311 127.0.0.1            LAPDm    S, func=RR, N(R)=2
    313 127.0.0.1            LAPDm    I, N(R)=2, N(S)=2(DTAP) (MM) Authentication Request
    314 127.0.0.1            LAPDm    S, func=RR, N(R)=3
    323 127.0.0.1            LAPDm    I, N(R)=3, N(S)=2(DTAP) (MM) Authentication Response
    329 127.0.0.1            LAPDm    I, N(R)=3, N(S)=3(DTAP) (RR) Ciphering Mode Command
    330 127.0.0.1            LAPDm    S, func=RR, N(R)=4
    331 127.0.0.1            LAPDm    I, N(R)=4, N(S)=3(DTAP) (RR) Ciphering Mode Complete up from here everything in this
                                                                   RR-connection is enciphered
    333 127.0.0.1            LAPDm    I, N(R)=4, N(S)=4(DTAP) (MM) Identity Request
    334 127.0.0.1            LAPDm    S, func=RR, N(R)=5
    335 127.0.0.1            LAPDm    I, N(R)=5, N(S)=4(DTAP) (MM) Identity Response
    339 127.0.0.1            LAPDm    I, N(R)=5, N(S)=5(DTAP) (MM) Location Updating Accept MS gets its TMSI here
    340 127.0.0.1            LAPDm    S, func=RR, N(R)=6
    341 127.0.0.1            LAPDm    I, N(R)=6, N(S)=5(DTAP) (MM) TMSI Reallocation Complete TMSI has been changed in MS
    348 127.0.0.1            LAPDm    I, N(R)=5, N(S)=6 (Fragment) some LAPDm frames are too big and will be fragmented
    349 127.0.0.1            LAPDm    S, func=RR, N(R)=7
    351 127.0.0.1            LAPDm    S, func=RR, N(R)=6
    353 127.0.0.1            LAPDm    I P, N(R)=6, N(S)=6 (Fragment)
    354 127.0.0.1            LAPDm    S F, func=REJ, N(R)=7
    355 127.0.0.1            LAPDm    I, N(R)=6, N(S)=7(DTAP) (MM) MM Information
    356 127.0.0.1            LAPDm    S, func=RR, N(R)=0
    357 127.0.0.1            LAPDm    I, N(R)=6, N(S)=0(DTAP) (RR) Channel Release
    358 127.0.0.1            LAPDm    S, func=RR, N(R)=1
    359 127.0.0.1            LAPDm    U P, func=DISC
    362 127.0.0.1            LAPDm    U F, func=UA
   2987 127.0.0.1            LAPDm    U P, func=SABM(DTAP) (MM) CM Service Request this RR-connection is for sending the SMS
   2992 127.0.0.1            LAPDm    U F, func=UA(DTAP) (MM) CM Service Request
   2994 127.0.0.1            LAPDm    I, N(R)=0, N(S)=0(DTAP) (MM) Identity Request
   2995 127.0.0.1            LAPDm    S, func=RR, N(R)=1
   2996 127.0.0.1            LAPDm    I, N(R)=1, N(S)=0(DTAP) (MM) Identity Response
   2997 127.0.0.1            LAPDm    I, N(R)=0, N(S)=1(DTAP) (MM) Authentication Request that e-plus BTS wanted the ms to
                                                                   authenticate again
   2998 127.0.0.1            LAPDm    S, func=RR, N(R)=2
   3006 127.0.0.1            LAPDm    S, func=RR, N(R)=1
   3007 127.0.0.1            LAPDm    I, N(R)=2, N(S)=1(DTAP) (MM) Authentication Response
   3008 127.0.0.1            LAPDm    I P, N(R)=1, N(S)=1(DTAP) (MM) Authentication Request
   3009 127.0.0.1            LAPDm    S F, func=REJ, N(R)=2
   3011 127.0.0.1            LAPDm    S, func=RR, N(R)=2
   3012 127.0.0.1            LAPDm    I, N(R)=2, N(S)=2(DTAP) (RR) Ciphering Mode Command
   3013 127.0.0.1            LAPDm    S, func=RR, N(R)=3
   3014 127.0.0.1            LAPDm    I, N(R)=3, N(S)=2(DTAP) (RR) Ciphering Mode Complete
   3015 127.0.0.1            LAPDm    U P, func=SABM
```

```
3020 127.0.0.1          LAPDm    S, func=RR, N(R)=3
3022 127.0.0.1          LAPDm    U F, func=UA
3023 127.0.0.1          LAPDm    I, N(R)=0, N(S)=0 (Fragment) that is the first SMS fragment, it is too big to fit in one frame
3024 127.0.0.1          LAPDm    S, func=RR, N(R)=1
3025 127.0.0.1          LAPDm    I, N(R)=0, N(S)=1 (Fragment) second fragment
3028 127.0.0.1          LAPDm    S, func=RR, N(R)=2
3029 127.0.0.1          GSM SMS  I, N(R)=0, N(S)=2(DTAP) (SMS) CP-DATA (RP) RP-DATA (MS to Network) and the complete SMS on an
                                                                                                     upper layer
3030 127.0.0.1          LAPDm    I, N(R)=3, N(S)=0(DTAP) (SMS) CP-ACK
3031 127.0.0.1          LAPDm    S, func=RR, N(R)=1
3033 127.0.0.1          GSM SMS  I, N(R)=3, N(S)=1(DTAP) (SMS) CP-DATA (RP) RP-ACK (Network to MS)
3036 127.0.0.1          LAPDm    S, func=RR, N(R)=2
3037 127.0.0.1          LAPDm    I, N(R)=2, N(S)=3(DTAP) (SMS) CP-ACK
3040 127.0.0.1          LAPDm    I, N(R)=3, N(S)=3(DTAP) (RR) Channel Release
3041 127.0.0.1          LAPDm    S, func=RR, N(R)=4
3042 127.0.0.1          LAPDm    U P, func=DISC
3044 127.0.0.1          LAPDm    S, func=RR, N(R)=4
3045 127.0.0.1          LAPDm    U F, func=UA
4630 127.0.0.1          LAPDm    U P, func=SABM(DTAP) (MM) CM Service Request this RR-connection is for an outgoing call
4635 127.0.0.1          LAPDm    U F, func=UA(DTAP) (MM) CM Service Request
4637 127.0.0.1          LAPDm    I, N(R)=0, N(S)=0(DTAP) (MM) Identity Request
4638 127.0.0.1          LAPDm    S, func=RR, N(R)=1
4639 127.0.0.1          LAPDm    I, N(R)=1, N(S)=0(DTAP) (MM) Identity Response no authentication this time, the BTS accepts the
                                                                                  ms's security context
4640 127.0.0.1          LAPDm    I, N(R)=0, N(S)=1(DTAP) (RR) Ciphering Mode Command
4641 127.0.0.1          LAPDm    S, func=RR, N(R)=2
4644 127.0.0.1          LAPDm    S, func=RR, N(R)=1
4645 127.0.0.1          LAPDm    I, N(R)=2, N(S)=1(DTAP) (RR) Ciphering Mode Complete
4646 127.0.0.1          LAPDm    I P, N(R)=1, N(S)=1(DTAP) (RR) Ciphering Mode Command
4647 127.0.0.1          LAPDm    S F, func=REJ, N(R)=2
4649 127.0.0.1          LAPDm    S, func=RR, N(R)=2
4650 127.0.0.1          LAPDm    I, N(R)=2, N(S)=2(DTAP) (CC) Setup the setup message with the called number
4654 127.0.0.1          LAPDm    I, N(R)=3, N(S)=2(DTAP) (CC) Call Proceeding
4657 127.0.0.1          LAPDm    S, func=RR, N(R)=3
4658 127.0.0.1          LAPDm/GSM MAP I, N(R)=3, N(S)=3(DTAP) (CC) Facility (GSM MAP) invoke notifySS
4659 127.0.0.1          LAPDm    S, func=RR, N(R)=4
4661 127.0.0.1          LAPDm    I, N(R)=3, N(S)=4 (Fragment)
4662 127.0.0.1          LAPDm    S, func=RR, N(R)=5
4663 127.0.0.1          LAPDm    I, N(R)=3, N(S)=5(DTAP) (RR) Assignment Command early assignment is configured, network wants
                                                                                    the MS to change to a TCH here
4664 127.0.0.1          LAPDm    S, func=RR, N(R)=6
4666 127.0.0.1          LAPDm    U P, func=SABM
4667 127.0.0.1          LAPDm    U F, func=UA
4668 127.0.0.1          LAPDm    I, N(R)=0, N(S)=0(DTAP) (RR) Assignment Complete this msg is already on TCH
4669 127.0.0.1          LAPDm    S, func=RR, N(R)=1
4670 127.0.0.1          LAPDm    I, N(R)=1, N(S)=0(DTAP) (CC) Progress
4671 127.0.0.1          LAPDm    S, func=RR, N(R)=1
4693 127.0.0.1          LAPDm    I, N(R)=1, N(S)=1(DTAP) (CC) Progress
4694 127.0.0.1          LAPDm    S, func=RR, N(R)=2
4695 127.0.0.1          LAPDm    I, N(R)=1, N(S)=2(DTAP) (CC) Alerting the called subscribers MS is ringing
4698 127.0.0.1          LAPDm    S, func=RR, N(R)=3
4732 127.0.0.1          LAPDm    I, N(R)=1, N(S)=3(DTAP) (CC) Connect the called subscriber took up the call
4735 127.0.0.1          LAPDm    S, func=RR, N(R)=4
4736 127.0.0.1          LAPDm    I, N(R)=4, N(S)=1(DTAP) (CC) Connect Acknowledge
4737 127.0.0.1          LAPDm    S, func=RR, N(R)=2
4777 127.0.0.1          LAPDm    I, N(R)=2, N(S)=4(DTAP) (CC) Disconnect somone hang up
4780 127.0.0.1          LAPDm    S, func=RR, N(R)=5
4781 127.0.0.1          LAPDm    I, N(R)=5, N(S)=2(DTAP) (CC) Release don't need TCH anymore, release it
4782 127.0.0.1          LAPDm    S, func=RR, N(R)=3
4783 127.0.0.1          LAPDm    I, N(R)=3, N(S)=5(DTAP) (CC) Release Complete
4784 127.0.0.1          LAPDm    S, func=RR, N(R)=6
4791 127.0.0.1          LAPDm    I, N(R)=3, N(S)=6(DTAP) (RR) Channel Release
4792 127.0.0.1          LAPDm    S, func=RR, N(R)=7
4793 127.0.0.1          LAPDm    U P, func=DISC
4801 127.0.0.1          LAPDm    U F, func=UA
```

Codebeispiel D.2: Aufgezeichneter Nachrichtenverkehr mit E-plus BTS, detailliert (mit LAPDm-Kontrollnachrichten), generiert mit Filtern aus Wireshark-Mitschnitt

```
No.     Destination          Protocol
    325 226.0.0.1            LAPDm    U P, func=SABM(DTAP) (MM) Location Updating Request
    326 225.0.0.1            LAPDm    U P, func=SABM(DTAP) (MM) Location Updating Request
    327 224.0.0.1            LAPDm    U F, func=UA(DTAP) (MM) Location Updating Request
    328 226.0.0.2            LAPDm    U F, func=UA(DTAP) (MM) Location Updating Request
    342 224.0.0.1            LAPDm    I, N(R)=0, N(S)=0(DTAP) (MM) Identity Request
    343 226.0.0.2            LAPDm    I, N(R)=0, N(S)=0(DTAP) (MM) Identity Request
    354 226.0.0.1            LAPDm    S, func=RR, N(R)=1
    355 225.0.0.1            LAPDm    S, func=RR, N(R)=1
    356 226.0.0.1            LAPDm    I, N(R)=1, N(S)=0(DTAP) (MM) Identity Response
    357 225.0.0.1            LAPDm    I, N(R)=1, N(S)=0(DTAP) (MM) Identity Response
    370 224.0.0.1            LAPDm    S, func=RR, N(R)=1
    371 226.0.0.2            LAPDm    S, func=RR, N(R)=1
    387 224.0.0.1            LAPDm    I, N(R)=1, N(S)=1(DTAP) (MM) Location Updating Accept
    388 226.0.0.2            LAPDm    I, N(R)=1, N(S)=1(DTAP) (MM) Location Updating Accept
    400 226.0.0.1            LAPDm    S, func=RR, N(R)=2
    401 225.0.0.1            LAPDm    S, func=RR, N(R)=2
```

```
    402 226.0.0.1             LAPDm    I, N(R)=2, N(S)=1(DTAP) (MM) TMSI Reallocation Complete
    403 225.0.0.1             LAPDm    I, N(R)=2, N(S)=1(DTAP) (MM) TMSI Reallocation Complete
    413 224.0.0.1             LAPDm    I, N(R)=2, N(S)=2 (Fragment)
    414 226.0.0.2             LAPDm    I, N(R)=2, N(S)=2 (Fragment)
    423 226.0.0.1             LAPDm    S, func=RR, N(R)=3
    424 225.0.0.1             LAPDm    S, func=RR, N(R)=3
    428 224.0.0.1             LAPDm    I, N(R)=2, N(S)=3(DTAP) (MM) MM Information
    429 226.0.0.2             LAPDm    I, N(R)=2, N(S)=3(DTAP) (MM) MM Information
    443 226.0.0.1             LAPDm    S, func=RR, N(R)=4
    444 225.0.0.1             LAPDm    S, func=RR, N(R)=4
    445 224.0.0.1             LAPDm/RRLP I, N(R)=2, N(S)=4(DTAP) (RR) Application Information
    446 226.0.0.2             LAPDm/RRLP I, N(R)=2, N(S)=4(DTAP) (RR) Application Information
    455 226.0.0.1             LAPDm    S, func=RR, N(R)=5
    456 225.0.0.1             LAPDm    S, func=RR, N(R)=5
    462 224.0.0.1             LAPDm    I, N(R)=2, N(S)=5(DTAP) (RR) Channel Release
    463 226.0.0.2             LAPDm    I, N(R)=2, N(S)=5(DTAP) (RR) Channel Release
    472 226.0.0.1             LAPDm    S, func=RR, N(R)=6
    473 225.0.0.1             LAPDm    S, func=RR, N(R)=6
    474 226.0.0.1             LAPDm    U P, func=DISC
    475 225.0.0.1             LAPDm    U P, func=DISC
    527 226.0.0.1             LAPDm    U P, func=DISC
    528 225.0.0.1             LAPDm    U P, func=DISC
    794 226.0.0.1             LAPDm    U P, func=SABM(DTAP) (MM) CM Service Request
    795 225.0.0.1             LAPDm    U P, func=SABM(DTAP) (MM) CM Service Request
    805 224.0.0.1             LAPDm    U F, func=UA(DTAP) (MM) CM Service Request
    806 226.0.0.2             LAPDm    U F, func=UA(DTAP) (MM) CM Service Request
    811 224.0.0.1             LAPDm    I, N(R)=0, N(S)=0(DTAP) (MM) CM Service Accept
    812 226.0.0.2             LAPDm    I, N(R)=0, N(S)=0(DTAP) (MM) Identity Request
>>>> Manipulated Identity Request sent from IMSI-Catcher to victim's MS
GSM TAP Header, ARFCN: 666 (Downlink), TS: 7, Channel: FACCH/F (0)
    Version: 2
    Header Length: 16 bytes
    Payload Type: GSM Um (MS<->BTS) (1)
    Time Slot: 7
    ..00 0010 1001 1010 = ARFCN: 666
    .0.. .... .... .... = Uplink: 0
    Signal/Noise Ratio (dB): 63
    Signal Level (dBm): 63
    GSM Frame Number: 2220920
    Channel Type: FACCH/F (9)
    Antenna Number: 0
    Sub-Slot: 0
Link Access Procedure, Channel Dm (LAPDm)
    Address Field: 0x03
        .00. .... = LPD: Normal GSM (0)
        ...0 00.. = SAPI: RR/MM/CC (0)
        .... ..1. = C/R: 1
        .... ...1 = EA: Final octet (1)
    Control field: I, N(R)=0, N(S)=0 (0x00)
        000. .... = N(R): 0
        .... 000. = N(S): 0
        .... ...0 = Frame type: Information frame (0x00)
    Length Field: 0x0d
        0000 11.. = Length: 3
        .... ..0. = M: Last segment (0)
        .... ...1 = EL: Final octet (1)
GSM A-I/F DTAP - Identity Request
    Protocol Discriminator: Mobility Management messages (5)
        .... 0101 = Protocol discriminator: Mobility Management messages (0x05)
        0000 .... = Skip Indicator: No indication of selected PLMN (0)
    00.. .... = Sequence number: 0
    ..01 1000 = DTAP Mobility Management Message Type: Identity Request (0x18)
    0000 .... = Spare bit(s): 0
    Identity Type
        .... 0... = Spare bit(s): 0
        .... .001 = Type of identity: IMSI (1)
>>>>
    815 226.0.0.1             LAPDm    S, func=RR, N(R)=1
    816 225.0.0.1             LAPDm    S, func=RR, N(R)=1
    817 226.0.0.1             LAPDm    I, N(R)=1, N(S)=0(DTAP) (MM) Identity Response
>>>> Identity Response from IMSI
    819 226.0.0.2             LAPDm    I, N(R)=1, N(S)=1(DTAP) (RR) Channel Release
    822 226.0.0.1             LAPDm    S, func=RR, N(R)=2
    823 225.0.0.1             LAPDm    S, func=RR, N(R)=2
    824 226.0.0.1             LAPDm    U P, func=DISC
    825 225.0.0.1             LAPDm    U P, func=DISC
    829 224.0.0.1             LAPDm    U F, func=UA
    830 226.0.0.2             LAPDm    U F, func=UA
   1466 226.0.0.1             LAPDm    U P, func=SABM(DTAP) (MM) CM Service Request
   1470 225.0.0.1             LAPDm    U P, func=SABM(DTAP) (MM) CM Service Request
   1479 224.0.0.1             LAPDm    U F, func=UA(DTAP) (MM) CM Service Request
   1480 226.0.0.2             LAPDm    U F, func=UA(DTAP) (MM) CM Service Request
   1483 224.0.0.1             LAPDm    I, N(R)=0, N(S)=0(DTAP) (MM) CM Service Accept
   1485 226.0.0.2             LAPDm    I, N(R)=0, N(S)=0(DTAP) (MM) CM Service Accept
   1491 226.0.0.1             LAPDm    S, func=RR, N(R)=1
   1493 226.0.0.1             LAPDm    I, N(R)=1, N(S)=0(DTAP) (CC) Setup
>>>> Original Setup Message sent from MS to Network and captured by MitM
GSM TAP Header, ARFCN: 666 (Uplink), TS: 6, Channel: FACCH/F (0)
```

```
    Version: 2
    Header Length: 16 bytes
    Payload Type: GSM Um (MS<->BTS) (1)
    Time Slot: 6
    ..00 0010 1001 1010 = ARFCN: 666
    .1.. .... .... .... = Uplink: 1
    Signal/Noise Ratio (dB): 63
    Signal Level (dBm): 63
    GSM Frame Number: 2221895
    Channel Type: FACCH/F (9)
    Antenna Number: 0
    Sub-Slot: 0
Link Access Procedure, Channel Dm (LAPDm)
    Address Field: 0x01
        .00. .... = LPD: Normal GSM (0)
        ...0 00.. = SAPI: RR/MM/CC (0)
        .... ..0. = C/R: 0
        .... ...1 = EA: Final octet (1)
    Control field: I, N(R)=1, N(S)=0 (0x20)
        001. .... = N(R): 1
        .... 000. = N(S): 0
        .... ...0 = Frame type: Information frame (0x00)
    Length Field: 0x51
        0101 00.. = Length: 20
        .... ..0. = M: Last segment (0)
        .... ...1 = EL: Final octet (1)
GSM A-I/F DTAP - Setup
    Protocol Discriminator: Call Control; call related SS messages (3)
        .... 0011 = Protocol discriminator: Call Control; call related SS messages (0x03)
        0... .... = TI flag: allocated by sender
        .000 .... = TIO: 0
    01.. .... = Sequence number: 1
    ..00 0101 = DTAP Call Control Message Type: Setup (0x05)
    Bearer Capability 1 - (MS supports at least full rate speech version 1 and half rate speech version 1. MS has a greater preference
                           for full rate speech version 1 than for half rate speech version 1)
        Element ID: 0x04
        Length: 4
        Octet 3
            0... .... = Extension: Extended
            .11. .... = Radio channel requirement: MS supports at least full rate speech version 1 and half rate speech version 1.
                        MS has a greater preference for full rate speech version 1 than for half rate speech version 1
            ...0 .... = Coding standard: GSM standardized coding
            .... 0... = Transfer mode: circuit
            .... .000 = Information transfer capability: Speech (0x00)
        Octets 3a - Speech Versions
            0... .... = Extension: Extended
            .0.. .... = Coding: octet used for extension of information transfer capability
            ..00 .... = Spare bit(s): 0
            .... 0010 = Speech version indication: GSM full rate speech version 2(GSM EFR) (0x02)
            0... .... = Extension: Extended
            .0.. .... = Coding: octet used for extension of information transfer capability
            ..00 .... = Spare bit(s): 0
            .... 0000 = Speech version indication: GSM full rate speech version 1(GSM FR) (0x00)
            1... .... = Extension: No Extension
            .0.. .... = Coding: octet used for extension of information transfer capability
            ..00 .... = Spare bit(s): 0
            .... 0001 = Speech version indication: GSM half rate speech version 1(GSM HR) (0x01)
    Called Party BCD Number - ()
        Element ID: 0x5e
        Length: 7
        1... .... = Extension: No Extension
        .000 .... = Type of number: unknown (0x00)
        .... 0001 = Numbering plan identification: ISDN/Telephony Numbering (ITU-T Rec. E.164 / ITU-T Rec. E.163) (0x01)
        Called Party BCD Number: 017518181818 displayed already decoded by wireshark
    Call Control Capabilities
        Element ID: 0x15
        Length: 1
        0000 .... = Maximum number of supported bearers: 1
        .... 0... = MCAT: The mobile station does not support Multimedia CAT
        .... .0.. = ENICM: The mobile station does not support the Enhanced Network-initiated In-Call Modification procedure
        .... ..0. = Prolonged Clearing Procedure: Not supported
        .... ...1 = DTMF: the mobile station supports DTMF as specified in subclause 5.5.7 of TS 24.008
>>>>
   1495 225.0.0.1             LAPDm    S, func=RR, N(R)=1
   1496 225.0.0.1             LAPDm    I, N(R)=1, N(S)=0(DTAP) (CC) Setup
>>>> Successfully manipulated Setup message sent from MitM to Network
GSM TAP Header, ARFCN: 666 (Uplink), TS: 6, Channel: FACCH/F (0)
    Version: 2
    Header Length: 16 bytes
    Payload Type: GSM Um (MS<->BTS) (1)
    Time Slot: 6
    ..00 0010 1001 1010 = ARFCN: 666
    .1.. .... .... .... = Uplink: 1
    Signal/Noise Ratio (dB): 63
    Signal Level (dBm): 63
    GSM Frame Number: 2221895
    Channel Type: FACCH/F (9)
    Antenna Number: 0
```

```
    Sub-Slot: 0
Link Access Procedure, Channel Dm (LAPDm)
    Address Field: 0x01
        .00. .... = LPD: Normal GSM (0)
        ...0 00.. = SAPI: RR/MM/CC (0)
        .... ..0. = C/R: 0
        .... ...1 = EA: Final octet (1)
    Control field: I, N(R)=1, N(S)=0 (0x20)
        001. .... = N(R): 1
        .... 000. = N(S): 0
        .... ...0 = Frame type: Information frame (0x00)
    Length Field: 0x51
        0101 00.. = Length: 20
        .... ..0. = M: Last segment (0)
        .... ...1 = EL: Final octet (1)
GSM A-I/F DTAP - Setup
    Protocol Discriminator: Call Control; call related SS messages (3)
        .... 0011 = Protocol discriminator: Call Control; call related SS messages (0x03)
        0... .... = TI flag: allocated by sender
        .000 .... = TIO: 0
    01.. .... = Sequence number: 1
    ..00 0101 = DTAP Call Control Message Type: Setup (0x05)
    Bearer Capability 1 - (MS supports at least full rate speech version 1 and half rate speech version 1. MS has a greater preference
                          for full rate speech version 1 than for half rate speech version 1)
        Element ID: 0x04
        Length: 4
        Octet 3
            0... .... = Extension: Extended
            .11. .... = Radio channel requirement: MS supports at least full rate speech version 1 and half rate speech version 1.
                        MS has a greater preference for full rate speech version 1 than for half rate speech version 1
            ...0 .... = Coding standard: GSM standardized coding
            .... 0... = Transfer mode: circuit
            .... .000 = Information transfer capability: Speech (0x00)
        Octets 3a - Speech Versions
            0... .... = Extension: Extended
            .0.. .... = Coding: octet used for extension of information transfer capability
            ..00 .... = Spare bit(s): 0
            .... 0010 = Speech version indication: GSM full rate speech version 2(GSM EFR) (0x02)
            0... .... = Extension: Extended
            .0.. .... = Coding: octet used for extension of information transfer capability
            ..00 .... = Spare bit(s): 0
            .... 0000 = Speech version indication: GSM full rate speech version 1(GSM FR) (0x00)
            1... .... = Extension: No Extension
            .0.. .... = Coding: octet used for extension of information transfer capability
            ..00 .... = Spare bit(s): 0
            .... 0001 = Speech version indication: GSM half rate speech version 1(GSM HR) (0x01)
    Called Party BCD Number - ()
        Element ID: 0x5e
        Length: 7
        1... .... = Extension: No Extension
        .000 .... = Type of number: unknown (0x00)
        .... 0001 = Numbering plan identification: ISDN/Telephony Numbering (ITU-T Rec. E.164 / ITU-T Rec. E.163) (0x01)
        Called Party BCD Number: 017517171717 successfully changed the number
    Call Control Capabilities
        Element ID: 0x15
        Length: 1
        0000 .... = Maximum number of supported bearers: 1
        .... 0... = MCAT: The mobile station does not support Multimedia CAT
        .... .0.. = ENICM: The mobile station does not support the Enhanced Network-initiated In-Call Modification procedure
        .... ..0. = Prolonged Clearing Procedure: Not supported
        .... ...1 = DTMF: the mobile station supports DTMF as specified in subclause 5.5.7 of TS 24.008
>>>>
   1508 224.0.0.1             LAPDm    S, func=RR, N(R)=1
   1509 226.0.0.2             LAPDm    S, func=RR, N(R)=1
   1512 224.0.0.1             LAPDm    I, N(R)=1, N(S)=1(DTAP) (CC) Call Proceeding
   1513 226.0.0.2             LAPDm    I, N(R)=1, N(S)=1(DTAP) (CC) Call Proceeding
   1520 226.0.0.1             LAPDm    S, func=RR, N(R)=2
   1522 225.0.0.1             LAPDm    S, func=RR, N(R)=2
   1532 224.0.0.1             LAPDm    I, N(R)=1, N(S)=2(DTAP) (RR) Channel Mode Modify
   1533 226.0.0.2             LAPDm    I, N(R)=1, N(S)=2(DTAP) (RR) Channel Mode Modify
   1538 226.0.0.1             LAPDm    S, func=RR, N(R)=3
   1542 225.0.0.1             LAPDm    S, func=RR, N(R)=3
   1544 226.0.0.1             LAPDm    I, N(R)=3, N(S)=1(DTAP) (RR) Channel Mode Modify Acknowledge
   1545 225.0.0.1             LAPDm    I, N(R)=3, N(S)=1(DTAP) (RR) Channel Mode Modify Acknowledge
   1561 224.0.0.1             LAPDm    I, N(R)=2, N(S)=3(DTAP) (CC) Release
   1562 226.0.0.2             LAPDm    I, N(R)=2, N(S)=3(DTAP) (CC) Release
   1569 226.0.0.1             LAPDm    S, func=RR, N(R)=4
   1570 225.0.0.1             LAPDm    S, func=RR, N(R)=4
   1571 226.0.0.1             LAPDm    I, N(R)=4, N(S)=2(DTAP) (CC) Release Complete
   1572 225.0.0.1             LAPDm    I, N(R)=4, N(S)=2(DTAP) (CC) Release Complete
   1580 224.0.0.1             LAPDm    S, func=RR, N(R)=3
   1581 226.0.0.2             LAPDm    S, func=RR, N(R)=3
   1587 224.0.0.1             LAPDm    I, N(R)=3, N(S)=4(DTAP) (RR) Channel Release
   1588 226.0.0.2             LAPDm    I, N(R)=3, N(S)=4(DTAP) (RR) Channel Release
   1593 226.0.0.1             LAPDm    S, func=RR, N(R)=5
   1594 225.0.0.1             LAPDm    S, func=RR, N(R)=5
   1595 226.0.0.1             LAPDm    U P, func=DISC
   1596 225.0.0.1             LAPDm    U P, func=DISC
```

```
1743 226.0.0.1            LAPDm    U P, func=DISC
1744 225.0.0.1            LAPDm    U P, func=DISC
```

Codebeispiel D.3: Erfolgreicher MitM-Angriff auf virtuellem Um, detailliert (mit LAPDm-Kontrollnachrichten), generiert mit Filtern aus Wireshark-Mitschnitt

Abkürzungsverzeichnis

3GPP	3rd Generation Partnership Project
A3	GSM Authentifizierungsalgorithmus
A5	Satz symmetrischer Verschlüsselungsalgorithmen für GSM
A8	GSM Algorithmus zur Schlüsselgenerierung
ACK	Acknowledgement
AGCH	Access Grant Channel
AKA	Authentication and Key Agreement
AK	Anonymity Key
AMF	Authentication and Key Management Field
APDU	Application Protocol Data Unit
APN	Access Point Name
ARFCN	Absolute Radio Frequency Channel Number
AuC	Authentication Center
AUTN	Authentication Vector Network
BCCH	Broadcast Control Channel
BCD	Binary Coded Decimal
BSC	Base Station Controller

BSS	Base Station Subsystem
BSSMAP	Base Station Subsystem Management Application Part
BTS	Base Transceiver Station
BTSM	Base Transceiver Station Management
CCBS	Completion of Calls to Busy Subscriber
CC	Call Control
CCCH	Common Control Channel
CC	Country Code
CEPT	European Conference of Postal and Telecommunications Administrations
CK	3G Cipher Key
CKSN	Cipher Key Sequence Number
CLI	Command Line Interface
CM	Connection Management
COMP128	Implementierung der GSM Sicherheitsalgorithmen A3 und A8
CON	Confirm
CRC	Cyclic Redundancy Check
C/R	Command/Response Bit
DCCH	Dedicated Control Channel
DECT	Digital Enhanced Cordless Telecommunications
DISC	Disconnect
DLCI	Data Link Connection Identifier
DSL	Digital Subscriber Line

DSP	Digital Signal Processor
EA	End of Address Bit
EDGE	Enhanced Data Rates for GSM Evolution
EGPRS	Enhanced GPRS
EIR	Equipment Identity Register
EL	End of Length Bit
ESP	Ecapsulated Security Payload
ETSI	European Telecommunications Standards Institute
FACCH	Fast Associated Control Channel
FCCH	Frequency Correction Channel
FDMA	Frequency Division Multiple Access
FN	Frame Number
GERAN	GSM EDGE Radio Access Network
GGSN	Gateway GPRS Support Node
GMSC	Gateway MSC
GMSK	Gaussion filtered Minimum Shift Keying
GPRS	General Packet Radio Service
GPU	Graphics Processing Unit
GSM	Global System for Mobile Communications
GSMTAP	GSM Test Access Protocol
HLR	Home Location Register
HSCSD	High Speed Circuit Switched Data

HSN Hopping Sequence Number

HSPA High Speed Packet Access

HTTP Hypertext Transfer Protocol

IANA Internet Assigned Numbers Authority

IEI Information Element Identifier

IK Integrity Key

IMEI International Mobile Equipment Identity

IMSI International Mobile Subscriber Identity

IND Indication

I Numbered Information Transfer Format

IoT Internet of Things

IP Internet Protocol

IPSec Internet Protocol Security

ISDN Integrated Services Digital Network

JVM Java Virtual Machine

Kc GSM Cipher Key

K Geheimer Schlüssel in 3G

Ki Geheimer Schlüssel in GSM

L1CTL Layer 1 Control

L2ML Layer 2 Management Link

LAI Location Area Index

LAPD Link Access Procedure for the D-Channel

LAPDm	Link Access Procedure for the Dm-Channel
LAU	Location Area Update
L	Length Indicator
LPD	Link Protocol Discriminator
LTE	Long Term Evolution
M2M	Machine to Machine
MAC	Message Authentication Code
MAIO	Mobile Allocation Index Offset
MA	Mobile Allocation
MAP	Mobile Application Part
ME	Mobile Equipment
MitM	Man-in-the-Middle
MM	Mobility Management
M	More Bit
MSC	Mobile Switching Center
MSISDN	Mobile Subscriber ISDN Number
MS	Mobile Station
MSRN	Mobile Subscriber Routing Number
MTP	Message Transfer Protocol
NB	Normal Burst
NDC	Network Destination Code
NITB	Network in the Box

N(R) Number of Received packets

NSA National Security Agency

N(S) Number of Sent packets

NSS Network Switching Subsystem

OML Operation and Maintenance Link

OSI Open Systems Interconnection

OTA Over the Air

PCH Paging Channel

PCU Packet Control Unit

P/F Poll/Final Bit

PLMN Public Land Mobile Network

RACH Random Access Channel

RAND Random Challenge in GSM Authentifizierung

REJ Reject

REQ Request

RES Response

RES Subscriber Response in GSM Authentifizierung

RR Receive Ready (LAPDm)

RR Radio Resource

RSL Radio Signaling Link

RSSI Received Signal Strength Indication

SABM Set Asynchronous Balanced Mode

SACCH Slow Associated Control Channel

SAPI SAP Identifier

SAP Service Access Point

SCCP Signaling Connection Control Part

SCH Synchronization Channel

SDCCH Standalone Dedicated Control Channel

SDR Software Defined Radio

SGSN Service GPRS Support Node

SIM Subscriber Identity Module

SI System Information

SMS Short Message Service

SN Subscriber Number

SQN Sequence Number

SRES Erwartete Subscriber Response in GSM Authentifizierung

SS7 Signaling System 7

SS Supplementary Services

S Supervisory Format

TAP Test Access Protocol

TA Timing Advance

TCH Traffic Channel

TDMA Time Division Multiple Access

TEI Terminal Endpoint Identifier

TETRA	Terrestrial Trunked Radio
TI	Transaction Identifier
TKG	Telekommunikationsgesetz
TLV	Type/Length/Value
TMSI	Temporary IMSI
TRAU	Transcoding and Rate Adaption Unit
UA	Unnumbered Acknowledgement
UDP	User Datagram Protocol
UI	Unumbered Information
Um	GSM Funkschnittstelle
UMTS	Universal Mobile Telecommunications System
USIM	UMTS Subscriber Identity Module
USRP	Universial Software Radio Peripheral
UTRAN	UMTS Terrestrial Radio Access Network
U	Unnumbered Information Transfer Format
VLR	Visitor Location Register
VoIP	Voice over IP
VoLTE	Voice over LTE
VTY	Virtual Teletype
WEP	Wired Equivalent Privacy
XMAC	Expected MAC
XOR	exklusives oder

XRES Expected Subscriber Response

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis für Code und Textbeispiele

Literaturverzeichnis

Wissenschaftliche Artikel

[Barkan u. a. 2003] Barkan, Elad ; Biham, Eli ; Keller, Nathan: Instant ciphertext-only cryptanalysis of GSM encrypted communication. In: *Annual International Cryptology Conference* Springer (Veranst.), 2003, S. 600–616 5.2, 8.1, 8.2, 8.3, B.3.3

[Bittau u. a. 2006] Bittau, Andrea ; Handley, Mark ; Lackey, Joshua: The final nail in WEP's coffin. In: *Security and Privacy, 2006 IEEE Symposium on* IEEE (Veranst.), 2006, S. 15–pp 5.1, 8.2, 8.3, 9

[Borisov u. a. 2001] Borisov, Nikita ; Goldberg, Ian ; Wagner, David: Intercepting mobile communications: the insecurity of 802.11. In: *Proceedings of the 7th annual international conference on Mobile computing and networking* ACM (Veranst.), 2001, S. 180–189 3.6.4

[Briceno u. a. 1998] Briceno, Marc ; Goldberg, Ian ; Wagner, David: *An implementation of the GSM A3A8 algorithm.* 1998 3.6

[Briceno u. a. 1999] Briceno, Marc ; Goldberg, Ian ; Wagner, David: A pedagogical implementation of the GSM A5/1 and A5/2 "voice privacy" encryption algorithms. In: *Originally published at http://www. scard. org, mirror at http://cryptome. org/gsm-a512. htm* (1999) 3.6

[Degabriele und Paterson 2007] Degabriele, Jean P. ; Paterson, Kenneth G.: Attacking the IPsec standards in encryption-only configurations. In: *Security and Privacy, 2007. SP'07. IEEE Symposium on* IEEE (Veranst.), 2007, S. 335–349 5.1, 8.2, 9

[Dunkelman u. a. 2010] Dunkelman, Orr ; Keller, Nathan ; Shamir, Adi: A Practical-Time Attack on the A5/3 Cryptosystem Used in Third Generation GSM Telephony. In: *IACR Cryptology ePrint Archive* 2010 (2010), S. 13 8.1, B.3.4

[Ericsson 2016] Ericsson: Cellular Networks for Massive IoT. (2016),

Nr. Uen:284-23-3278. – URL `http://www.ericsson.com/res/docs/whitepapers/wp_iot.pdf` 1

[Fox 2002] Fox, Dirk: Der IMSI-catcher. In: *Datenschutz und Datensicherheit* 26 (2002), Nr. 4, S. 212–215 8.2

[Goldberg u. a. 1999] Goldberg, Ian ; Wagner, David ; Green, Lucky: The real-time cryptanalysis of A5/2. In: *Rump session of Crypto* 99 (1999), S. 239–255 8.1

[Golde u. a. 2012] Golde, Nico ; Redon, Kevin ; Borgaonkar, Ravishankar: Weaponizing Femtocells: The Effect of Rogue Devices on Mobile Telecommunications. In: *NDSS*, 2012 8.2

[Golić 1997] Golić, Jovan D.: Cryptanalysis of alleged A5 stream cipher. In: *International Conference on the Theory and Applications of Cryptographic Techniques* Springer (Veranst.), 1997, S. 239–255 8.1, B.3.2

[Hulton 2008] Hulton, David: Intercepting GSM traffic. In: *BlackHat Briefings* (2008) 8.1

[McMilin und Ali 2017] McMilin, Emily ; Ali, Kashif: OpenCellular: Open Source Wireless Access Platform, URL `https://osmocom.org/attachments/download/2623/opencellular.pdf`, 04 2017 4

[Meyer und Wetzel 2004] Meyer, Ulrike ; Wetzel, Susanne: A man-in-the-middle attack on UMTS. In: *Proceedings of the 3rd ACM workshop on Wireless security* ACM (Veranst.), 2004, S. 90–97 (document), 5.2, 8.2, 8.3, 9

[Mourad 2015] Mourad, Hassan: The Fall of SS7 - How Can the Critical Security Controls Help. (2015) 8.2

[Nohl 2013] Nohl, Karsten: Rooting SIM cards. In: *Black Hat USA* 2013 (2013) 8.2

[Nohl 2014] Nohl, Karsten: Mobile self-defense. In: *Vortrag auf dem Chaos Communication Congress 31C3, Hamburg*, 2014 8.1, 8.2, B.3.4

[Nohl und Munaut 2010] Nohl, Karsten ; Munaut, Sylvain: Wideband GSM sniffing. In: *27th Chaos Communication Congress*, 2010 8.1, B.3.2

[Nohl und Paget 2009] Nohl, Karsten ; Paget, Chris: Gsm: Srsly. In: *26th Chaos Communication Congress* Bd. 8, 2009, S. 11–17 8.1, 8.2

[Nomine 2017] Nomine, Roch-Alexandre: Showcase: Running a commercial cellular

network with OsmoBTS/OsmoPCU/OsmoBSC & co, URL `https://osmocom.org/attachments/download/2617/commercial_network_osmocom.pdf`, 04 2017 4

[Paget 2010] Paget, Chris: Practical cellphone spying. In: *Def Con* 18 (2010) 5.2, 8.2

[Paterson und Yau 2006] Paterson, Kenneth G. ; Yau, Arnold K.: Cryptography in theory and practice: The case of encryption in IPsec. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques* Springer (Veranst.), 2006, S. 12–29 5.1, 8.2, 9

[Ramadan 2017] Ramadan, Omar: Community Cellular Manager, URL `https://media.ccc.de/v/osmocon17-4014-community_cellular_manager`, 04 2017 4

[Solnik und Blanchou 2014] Solnik, M ; Blanchou, M: Cellular exploitation on a global scale: The rise and fall of the control protocol. In: *Black Hat USA* (2014) 8.2

[Yu u. a. 2004] Yu, Tom ; Hartman, Sam ; Raeburn, Kenneth: The Perils of Unauthenticated Encryption: Kerberos Version 4. In: *NDSS* Bd. 4, 2004, S. 4–4 8.2, 9

[Zou] Zou, Alex: DIGITAL SIGNAL PROCESSING IN IF/RF DATA CONVERTERS. 3.12, 3.5

Bücher

[Biryukov u. a. 2001] Biryukov, Alex ; Shamir, Adi ; Wagner, David: *Real Time Cryptanalysis of A5/1 on a PC*. S. 1–18. In: Goos, Gerhard (Hrsg.) ; Hartmanis, Juris (Hrsg.) ; Leeuwen, Jan van (Hrsg.) ; Schneier, Bruce (Hrsg.): *Fast Software Encryption: 7th International Workshop, FSE 2000 New York, NY, USA, April 10–12, 2000 Proceedings*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2001. – URL `http://dx.doi.org/10.1007/3-540-44706-7_1`. – ISBN 978-3-540-44706-1 8.1

[Eberspächer u. a. 2008] Eberspächer, Jörg ; Vögel, Hans-Jörg ; Bettstetter, Christian ; Hartmann, Christian: *GSM – Architecture, Protocols and Services*. John Wiley & Sons, Ltd, 2008. – URL `http://dx.doi.org/10.1002/9780470741719.ch1`. – ISBN 9780470741719 3.2, 3.2, 3.5, 3.5.2, B.1

[Fire 1959] Fire, Philip: *A class of multiple-error-correcting binary codes for non-independent errors*. Bd. 55. Department of Electrical Engineering, Stanford University., 1959 3.5.1

[Göbel u. a. 1996] Göbel, Klaus ; Leiss, Ludwig ; Marquardt, Klaus: *Strafprozess*. S. 46, Beck, 1996 8.2

[Sauter 2011] Sauter, Martin: *Grundkurs Mobile Kommunikationssysteme*. Vieweg + Teubinger, 2011. – URL `http://www.springer.com/de/book/9783658083410`. – ISBN 9783834814074 3.4, 3.5

[Schnabel 2003] Schnabel, P.: *Kommunikationstechnik-Fibel: Grundlagen, Festnetz, Mobilfunktechnik, Breitbandtechnik, Netzwerktechnik*. Schnabel, 2003. – URL `http://www.elektronik-kompendium.de/`. – ISBN 9783833005671 3.1, B.2.1, B.2.1

[Werner 2008] Werner, Martin: *Information und Codierung: Grundlagen und Anwendungen*. Berlin, Heidelberg : Springer-Verlag, 2008. – ISBN 978-3-8348-9550-9 3.5.1, 3.5.2, 5.7, 5.7

Internet

[3gpp.org 2017a] 3gpp.org: *GSM Spec history*. 2017. – URL `http://www.3gpp.org/specifications/gsm-history`. – Zugriffsdatum: [05. Juni 2017] A

[3gpp.org 2017b] 3gpp.org: *Releases*. 2017. – URL `http://www.3gpp.org/specifications/releases`. – Zugriffsdatum: [05. Juni 2017] A

[ftp.osmocom.org 2017] ftp.osmocom.org: *Latest user documentations*. 2017. – URL `http://ftp.osmocom.org/docs/latest/`. – Zugriffsdatum: [05. Juni 2017] 4.2, 4.3

[gsmmap.org 2016] gsmmap.org: *Mobile Network Security report: Germany*. 08 2016. – URL `https://gsmmap.org/assets/pdfs/gsmmap.org-country_report-Germany-2016-08.pdf`. – Zugriffsdatum: [05. Juni 2017] 3.6.4, 3.4, 8.1, 8.2, B.3.2

[handy-flatrate-24.de 2017] handy-flatrate-24.de: *UMTS, GPR-S, LTE & Co. – Übertragungstechniken im Mobilfunk*. 2017. – URL `https://www.handy-flatrate-24.de/informationen/umts-gprs-lte-und-co-uebertragungstechniken-im-mobilfunk.php`. – Zugriffsdatum: [05. Juni 2017] A

[heise.de 2016] heise.de: *GSM-Zukunft in Deutschland, Österreich und der Schweiz ist offen*. 12 2016. – URL `https://www.heise.de/newsticker/meldung/GSM-Zukunft-in-Deutschland-Oesterreich-und-der-Schweiz-ist-offen-3582914.html`. – Zugriffsdatum: [05. Juni 2017] 1

[laforge.gnumonks.org 2010] LAFORGE.GNUMONKS.ORG: *A brief history on the withdrawal of the A5/2 ciphering algorithm in GSM*. 11 2010. – URL `http://laforge.gnumonks.org/blog/20101112-history_of_a52_withdrawal/`. – Zugriffsdatum: [05. Juni 2017] 1

[lte-anbieter.info 2017a] LTE-ANBIETER.INFO: *5G: Alles zum LTE-Nachfolger der Zukunft*. 2017. – URL `http://www.lte-anbieter.info/5g/`. – Zugriffsdatum: [05. Juni 2017] A

[lte-anbieter.info 2017b] LTE-ANBIETER.INFO: *VoLTE: Voice over LTE - Alles über echte Telefonie via 4G Mobilfunknetze*. 2017. – URL `http://www.lte-anbieter.info/volte/`. – Zugriffsdatum: [05. Juni 2017] A

[ltemap.de 2017] LTEMAP.DE: *LTE Karte: Netzabdeckung von Vodafone, O2, Telekom und E-Plus*. 2017. – URL `http://www.ltemap.de/`. – Zugriffsdatum: [05. Juni 2017] A

[mobileworldlive.com 2015] MOBILEWORLDLIVE.COM: *Operators and Vendors predict long life for 2G*. 06 2015. – URL `http://www.mobileworldlive.com/featured-content/top-three/operators-vendors-forecast-long-life-2g/`. – Zugriffsdatum: [05. Juni 2017] 1

[opensignal.com 2017] OPENSIGNAL.COM: *Global Cell Coverage Maps*. 2017. – URL `https://opensignal.com/networks/`. – Zugriffsdatum: [05. Juni 2017] 1, A

[osmocom.org 2017a] OSMOCOM.ORG: *Cellular Infrastructure - Nightly Builds*. 2017. – URL `https://osmocom.org/projects/cellular-infrastructure/wiki/Nightly_Builds`. – Zugriffsdatum: [05. Juni 2017] 4

[osmocom.org 2017b] OSMOCOM.ORG: *Cellular Infrastructure: Rhizomatica hackathon on rural GSM based on Osmocom*. 2017. – URL `https://osmocom.org/news?page=3`. – Zugriffsdatum: [05. Juni 2017] 1, 4

[osmocom.org 2017c] OSMOCOM.ORG: *Open Source Mobile Communications*. 2017. – URL `https://osmocom.org/`. – Zugriffsdatum: [05. Juni 2017] 4

[osmocom.org 2017d] OSMOCOM.ORG: *Osmocom libraries - libosmocore*. 2017. – URL `https://osmocom.org/projects/libosmocore/wiki/Libosmocore`. – Zugriffsdatum: [05. Juni 2017] 4

[osmocom.org 2017e] OSMOCOM.ORG: *Project Rationale*. 2017. – URL `https://osmocom.org/projects/baseband/wiki/ProjectRationale`. – Zugriffsdatum: [05. Juni 2017] 4

[osmocom.org 2017f] OSMOCOM.ORG: *Welcome to the OsmocomBB project*. 2017. – URL `https://osmocom.org/projects/baseband/wiki`. – Zugriffsdatum: [05. Juni 2017] 4.1

[security.osmocom.org 2017] SECURITY.OSMOCOM.ORG: *Withdrawal of A5/2 algorithim support*. 2017. – URL `http://security.osmocom.org/trac/wiki/A52_Withdrawal`. – Zugriffsdatum: [05. Juni 2017] 8.1, B.3.3

[telekom.com 2013] TELEKOM.COM: *Telekom erhöht Abhörschutz im Mobilfunk*. 12 2013. – URL `https://www.telekom.com/de/medien/medieninformationen/detail/telekom-erhoeht-abhoerschutz-im-mobilfunk-343760`. – Zugriffsdatum: [05. Juni 2017] B.3.4

[teltarif.de 2017] TELTARIF.DE: *GSM-Repeater: Funkloch selbst stopfen - die Rechtslage*. 2017. – URL `https://www.teltarif.de/repeater-funkloch-genehmigung/news/53267.html`. – Zugriffsdatum: [05. Juni 2017] 8.3

[theintercept.com 2014] THEINTERCEPT.COM: *Operation Auroragold – How the NSA Hacks Cellphone Networks Worldwide*. 12 2014. – URL `https://theintercept.com/2014/12/04/nsa-auroragold-hack-cellphones/`. – Zugriffsdatum: [05. Juni 2017] 8.1

Technische Spezifikationen und Empfehlungen

[Bundesnetzagentur 2013] BUNDESNETZAGENTUR: Nummernplan Rufnummern für Mobile Dienste / Bundesnetzagentur. URL `https://www.bundesnetzagentur.de/DE/Sachgebiete/Telekommunikation/Unternehmen_Institutionen/Nummerierung/Rufnummern/MobileDienste/MobileDienste_Basepage.html`, September 2013 (Verfügung 11/2011, Amtsblatt 04/2011). – Verfügung 5.2

[TR-31.900 2012] TR-31.900: SIM/USIM internal and external interworking aspects / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/31900.htm`, September 2012 (31.900). – Technical Report 3.19

[TREC-E.164 2010a] TREC-E.164: List of recommendation ITU-T E.164 assigned country codes / International Telecommunication Union (ITU). URL `http://www.itu.int/rec/T-REC-E.164`, November 2010 (E.164). – Technical Recommendation 5.6

[TREC-E.164 2010b] TREC-E.164: The international public telecommunication numbering plan / International Telecommunication Union (ITU). URL `http://www.itu.int/rec/T-REC-E.164`, November 2010 (E.164). – Technical Recommendation 5.6

[TREC-G.703 2016] TREC-G.703: Physical/electrical characteristics of hierarchical digital interfaces / International Telecommunication Union (ITU). URL `https://www.itu.int/rec/T-REC-G.703`, August 2016 (G.703). – Technical Recommendation B.2.2

[TREC-I.330 1988] TREC-I.330: ISDN numbering and addressing principles / International Telecommunication Union (ITU). URL `https://www.itu.int/rec/T-REC-I.330`, November 1988 (I.330). – Technical Recommendation 5.5

[TREC-I.334 1988] TREC-I.334: Principles relating ISDN numbers/sub-addresses to the OSI reference model network layer addresses / International Telecommunication Union (ITU). URL `https://www.itu.int/rec/T-REC-I.334`, November 1988 (I.334). – Technical Recommendation 5.5

[TREC-Q.920 1994] TREC-Q.920: ISDN user-network interface data link layer - General aspects / International Telecommunication Union (ITU). URL `https://www.itu.int/rec/T-REC-Q.920`, Dezember 1994 (Q.920). – Technical Recommendation B.2.2

[TREC-Q.921 1998] TREC-Q.921: ISDN user-network interface - Data link layer specification / International Telecommunication Union (ITU). URL `https://www.itu.int/rec/T-REC-Q.921`, Juli 1998 (Q.921). – Technical Recommendation B.2.2

[TS-03.20 2007] TS-03.20: Security related network functions / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/0320.htm`, Dezember 2007 (03.20 - v8.6.0). – Technical Specification 3.6, 3.15, 3.6.1, 3.16, 3.6.2, 3.6.4, 3.20, A, B.3.1

[TS-04.03 2002] TS-04.03: Channel structures and access capabilities / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/0403.htm`, 2002 (04.03 - v8.0.2). – Technical Specification 3.3.1, 3.3.1

[TS-04.04 2002] TS-04.04: Layer 1 General requirements / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/0404.htm`, 2002 (04.04 - v8.1.2). – Technical Specification 3.3.2, 3.5, 3.6, 3.7

[TS-04.05 2002] TS-04.05: Data Link Layer LAPDm General Aspects / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/0405.htm`, 2002 (04.05 - v8.0.2). – Technical Specification 3.3.3, 3.3.3, 3.3.3

[TS-04.06 2008] TS-04.06: Data Link (DL) layer specification / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/0406.htm`, Dezember 2008 (04.06 - v8.4.0). – Technical Specification 3.3.3, 3.3.3, 3.3.3, 3.7, 3.3.3, 3.8, 3.9, 3.10, 5.1, D.1

[TS-04.18 2006] TS-04.18: Radio Resource Control Protocol / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/0418.htm`, 2006 (04.18 - v8.27.0). – Technical Specification 3.4, 3.4.1, 5.3, 5.3, 5.3, 6.5.1, D.1

[TS-05.02 2003] TS-05.02: Layer 1 Multiplexing and multiple access / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/0502.htm`, Juni 2003 (05.02 - v8.11.0). – Technical Specification 3.3, 3.3, 3.3.1, 5.4, B.2.1, B.1, B.2.1, B.4

[TS-05.03 2005] TS-05.03: Channel coding / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/0503.htm`, Januar 2005 (05.03 - v8.9.0). – Technical Specification 3.12, 3.13, 3.5, 3.5.1, 3.5.2, 3.5.2, 3.5.3, 3.5.3, 3.5.3, 3.5.4

[TS-05.04 2002] TS-05.04: Modulation / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/0504.htm`, Januar 2002 (05.04 - v8.4.0). – Technical Specification B.2.1

[TS-08.56 2003] TS-08.56: (BSC - BTS) interface; Layer 2 specification / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/0856.htm`, September 2003 (08.56 - v8.0.1). – Technical Specification B.2.2

[TS-08.58 2000] TS-08.58: (BSC - BTS) interface; Layer 3 specification / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/0858.htm`, November 2000 (08.58 - v8.6.0). – Technical Specification B.2.2

[TS-23.003 2017] TS-23.003: Numbering, addressing and identification / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/23003.htm`, März 2017 (23.003 - v14.3.0). – Technical Specification 5.6

[TS-23.108 2015] TS-23.108: Mobile radio interface layer 3 specification, Core network protocols; Stage 2 / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/23108.htm`, Dezember 2015 (23.018 - v13.0.0). – Technical Specification 3.4, 3.4.2, 5.3, 5.4, 5.3, 5.5, 5.6

[TS-24.007 2005] TS-24.007: Mobile radio interface signalling layer 3; General aspects / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/24007.htm`, September 2005 (24.007 - v7.0.0). – Technical Specification 3.4, 3.2, 3.3, 3.4.3, 5.3, 5.4, 5.5, 5.6, 5.6, D.1

[TS-24.008 2005] TS-24.008: Mobile radio interface layer 3 specification; Core Network Protocols; Stage 3 / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/24008.htm`, Dezember 2005 (24.008 - v5.15.0).

– Technical Specification 3.4, 3.11, 3.4.2, 3.4.3, 3.6.4, 3.6.4, 5.3, 5.3, 5.3, 5.3, 5.4, 5.7, 5.4, 5.5, 5.8, 5.5, 5.5, 5.9, 5.6, 5.3, 5.4, 5.6, 6.5.1, 8.2

[TS-24.010 2011] TS-24.010: Mobile radio interface layer 3; Supplementary services specification / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/24010.htm`, Juni 2011 (24.010 - v10.1.0). – Technical Specification 3.4.3

[TS-24.011 2005] TS-24.011: Point-to-Point (PP) Short Message Service (SMS) support on mobile radio interface / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/24011.htm`, Juni 2005 (24.011 - v6.1.0). – Technical Specification 3.4.3

[TS-24.080 2011] TS-24.080: Mobile radio interface layer 3 supplementary services specification; Formats and coding / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/24080.htm`, März 2011 (24.080 - v10.0.0). – Technical Specification 5.5

[TS-33.102 2016] TS-33.102: Security related network functions / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/33102.htm`, September 2016 (33.102 - v14.0.0). – Technical Specification 3.6.3, 3.17, 3.6.3, 3.18, 3.6.3, A

[TS-35.202 2016] TS-35.202: Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/35202.htm`, Januar 2016 (35.202 - v13.0.0). – Technical Specification B.3.4

[TS-55.216 2003] TS-55.216: Specification of the A5/3 algorithms for GSM / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/55216.htm`, September 2003 (55.216 - v6.2.0). – Technical Specification B.3.4

[TS-55.226 2011] TS-55.226: Specification of the A5/4 Encryption Algorithms for GSM and ECSD, and the GEA4 Encryption Algorithm for GPRS / 3rd Generation Partnership Project (3GPP). URL `http://www.3gpp.org/ftp/Specs/html-info/55226.htm`, Februar 2011 (55.226 - 10.0.0). – Technical Specification 3.6.4, 8.1, B.3.5, B.3.6

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Titel **Ein vom Verschlüsselungsverfahren unabhängiger Man-in-the-Middle Angriff in GSM / An encryption independent Man-In-The-Middle Attack in GSM** selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel verwendet sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

München, 06.06.2017

(Sebastian Stumpf)