RUHR-UNIVERSITÄT BOCHUM

Sicherheit Smartcard-basierter Zugangskontrollsysteme

Alexander Steffen

Master's Thesis. 16. Januar 2012.
Chair for Embedded Security – Prof. Dr.-Ing. Christof Paar

Zusammenfassung

Diese Arbeit gibt einen Überblick über Smartcard-Technologien, ihre Einsatzzwecke und mögliche Schwachstellen. Ein weit verbreitetes System, Legic Prime, wird im Detail vorgestellt und dessen vor einiger Zeit veröffentlichten Sicherheitslücken beschrieben. Auf dieser Basis folgt die Untersuchung des bislang undokumentierten Nachfolgers, Legic Advant. Hier scheinen zwar bei der Kartentechnologie selbst die Probleme ausgeräumt, dafür zeigen sich aber Unzulänglichkeiten im stationären Teil des überprüften Zugangskontrollsystems. Zudem werden die zur Untersuchung verwendeten Techniken dokumentiert, um so als Vorlage für die Analyse weiterer Systeme dienen zu können.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Alexander Steffen

Inhaltsverzeichnis

1 Einleitung

> Any sufficiently advanced technology is indistinguishable from magic.
>
> — Arthur C. Clarke *Profiles of the Future*

Smartcard-Technologien haben das Potential eines Tages so mit unserem Alltag zu verschmelzen, dass wir sie als solche nicht mehr wahrnehmen. Schon heute haben sie Einzug in verschiedenste Bereiche unseres Lebens gehalten, um dort viele Prozesse zu vereinfachen und den Komfort für die Benutzer zu erhöhen. Sie erlauben das bargeldlose Bezahlen, ermöglichen die unkomplizierte Übertragung von Informationen und gewähren Zugang zu Autos, Gebäuden und anderen Dingen.

Allerdings wird die Sicherheit dieser Anwendungen oft nur unzureichend gewährleistet, wie zahlreiche Angriffe in der Vergangenheit gezeigt haben. Dabei sind Smartcards als in Hardware gegossener Tresor für Informationen [1, S.11] insbesondere für sicherheitskritische Anwendungen prädestiniert. Sie können geheimes Schlüsselmaterial für sichere und authentifizierte Kommunikation speichern, oder einen weiteren Beitrag zur Fälschungssicherheit von Ausweisen leisten.

Angreifer konnten jedoch in vielen Fällen beworbene Eigenschaften außer Kraft setzen. Geschützte Informationen ließen sich auslesen oder manipulieren, „nicht-klonbare“ Karten konnten geklont werden. Viele der Systeme galten nur deshalb so lange als sicher, weil die Funktionsweise nicht öffentlich dokumentiert war und niemand sie genauer analysierte. Nur langsam kommt es zu einem Umdenken, das Design eines Systems nicht mehr völlig unter Verschluss zu halten, sodass sich auch unabhängige Stellen ein Bild von der Güte der Sicherheitsversprechen machen können.

Diese Arbeit soll einige der Sicherheitsprobleme am Beispiel von Zugangskontrollsystemen, wie sie etwa in Unternehmen zum Einsatz kommen, näher beleuchten und herausfinden, inwieweit bei moderneren Systemen dazugelernt wurde. Dazu erfolgt zunächst ein Überblick über die Entwicklung von Smartcards (Abschnitt 2), ihre vielfältigen Anwendungsgebiete (Abschnitt 2.1) und generelle Funktionsweise (Abschnitt 2.2), bevor einige der bekannteren Angriffe und mögliche Abwehrmaßnahmen vorgestellt werden (Abschnitt 2.3).

Die Funktionsweise und Schwachstellen der weit verbreiteten Legic-Prime-Karten werden detailliert dargestellt (Abschnitt 3), um die Grundlage für die Analyse des Nachfolgers Legic Advant, für den bislang noch keine öffentliche Untersuchung bekannt ist, zu legen (Abschnitt 4). Hier wird nicht nur die Funkschnittstelle, die beim Vorgänger bereits zur vollständigen Kompromittierung des Systems genügte, betrachtet (Abschnitt 4.1), sondern ein vollständiges Zugangskontrollsystem mitsamt den stationären Lesegeräten und deren Backend untersucht (Abschnitt 4.2). Um diese Analyse nachvollziehbar zu

machen und Möglichkeiten für die Analyse anderer Systeme aufzuzeigen, werden die dabei verwendeten Werkzeuge ausführlich dokumentiert (Abschnitt 5).

2 Smartcards

Die Geschichte der Smartcards beginnt in den fünfziger Jahren des letzten Jahrhunderts mit der Einführung von reinen Plastikkarten als Zahlungsmittel durch Diner's Club. Mit Vorlage dieser Karten konnte bei ausgewählten Partnern bargeldlos bezahlt werden [2, S.2].

Mit dem Einstieg weiterer Anbieter, den heutigen Unternehmen VISA und MasterCard, und der zunehmenden Verbreitung solcher Karten, stiegen sowohl die Kosten für die Verwaltung und Verwendung der Karten als auch die Schäden durch Betrugsversuche. Dem versuchte man mit der Einführung von maschinenlesbaren Karten mit Magnetstreifen, wie sie auch heute noch verwendet werden, abzuhelfen. Dies vereinfacht zwar die Handhabung der Karten, schützt allein jedoch kaum vor Betrug, denn die im Magnetstreifen gespeicherten Daten sind weder gegen Auslesen noch Verändern geschützt. Daher müssen alle Transaktionen zeitgleich von einem zentralen Backend-System bestätigt werden, um etwaige Manipulationen zu erkennen [3, S.1].

Da diese Infrastruktur nicht überall zur Verfügung stand, blieb als Alternative nur, die gespeicherten Daten auf der Karte besser zu schützen, indem man ihr eine eigene Logik in Form von integrierten Schaltkreisen spendierte. Die ersten Patente für solche Integrated Circuit Cards (ICC) oder Smartcards wurden 1968 von deutschen und in den folgenden Jahren auch von japanischen und französischen Entwicklern eingereicht [3, S.2].

2.1 Anwendungsfälle

Seit ihrer Entwicklung haben sich zahlreiche Einsatzgebiete für Smartcards ergeben. Diese lassen sich grob in die Kategorien Bezahlung, Authentifizierung und Identifizierung einteilen, für die im Folgenden einige Beispiele gegeben werden.

Dabei lassen sich drei grundsätzliche Motivationen für den Einsatz von Smartcards erkennen. Zunächst ermöglichen diese über standardisierte Schnittstellen die maschinenlesbare Speicherung von Informationen, was insbesondere bei der Ablösung Papier-basierter Prozesse Aufwand und Fehler reduziert. Darüber hinaus können mit den integrierten Schaltkreisen nur schwerlich zu umgehende Sicherheitsmechanismen implementiert werden, sodass unbefugtes Auslesen oder Verändern von Daten verhindert wird. Durch immer bessere Fertigungsverfahren, und dadurch steigende Rechenleistung und Speicherplatz bei sinkenden Kosten, bietet es sich schließlich an, mehrere zuvor getrennte Funktionen in einem Medium zusammenzufassen und so den Komfort für den Anwender zu erhöhen.

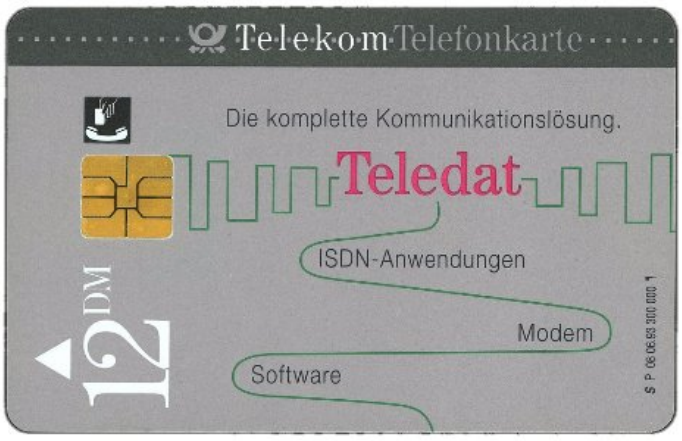

Abbildung 2.1: Telefon-Karte (basierend auf [7] Nightflyer / CC-BY-SA-3.0)

2.1.1 Bezahlung

Allen Anwendungen im Bereich Bezahlung ist gemein, dass Bargeld durch „Plastikgeld“ abgelöst wird, um den Komfort für alle Beteiligte zu erhöhen, da auf den Transport und die Sicherung der Bargeldbestände verzichtet werden kann. Zusätzlich können sich Bezahlvorgänge beschleunigen, wenn das Abzählen der korrekten Geldmenge entfällt. Mitunter wird das Guthaben oder der Verfügungsrahmen direkt auf der Karte gespeichert, um Bezahlvorgänge auch ohne ständige Verbindung zu einem Backend-System zu ermöglichen.

Öffentliche Telefone

Eine der ersten großen Anwendungen war die Ablösung von Münztelefonen durch bargeldlose Kartentelefone unter anderem in Deutschland und Frankreich, für die bereits Mitte der achtziger Jahre mehrere Millionen Telefonkarten (Abbildung 2.1) im Umlauf waren [3, S.2]. Kartentelefone haben für den Betreiber den Vorteil, dass keine regelmäßige Leerung erfolgen muss und weniger Anreiz für Beschädigungen, etwa um an den Münzvorrat zu gelangen, besteht. Probleme gab es allerdings mit einigen Varianten der Telefonkarten, deren Guthabensstand nachträglich wieder erhöht werden konnte [4].

Mit der Verbreitung von Mobiltelefonen ging die Bedeutung der Telefonkarten schnell zurück. So besitzen heutige öffentliche Telefone mitunter gar keinen Kartenschlitz mehr und akzeptieren zur Bezahlung nur Kreditkarten oder spezielle Guthaben-Karten (Calling-Cards), deren Guthaben nicht auf der Karte, sondern direkt beim Betreiber gespeichert ist [5][6].

Abbildung 2.2: Bank-Karte mit Chip und Magnetstreifen (Christian Horvat [8] / CC-BY-SA-3.0)

Banken

Auch Banken erkannten bald die Vorteile der Smartcards und gaben entsprechende Bankkarten an ihre Kunden heraus, in Frankreich beispielsweise bereits 1984, in Deutschland erst 1997 [3, S.2]. Heute ist es mit diesen Karten oft nicht nur möglich, Geld abzuheben und Kontoauszüge abzuholen, sondern es wurden weitere Bezahlmöglichkeiten integriert, etwa als Kredit- oder Geldkarte (Abbildung 2.2).

Kantinen

Private Unternehmen setzen ebenfalls häufig auf Smartcard-Technologien, um die Bezahlprozesse in eigenen Kantinen zu optimieren. Entweder werden bestehende bargeldlose Zahlsysteme wie die Geldkarte akzeptiert [9], oder ein beispielsweise schon für die Zu-

Abbildung 2.3: Ticket zur Fußball-Weltmeisterschaft 2006 (basierend auf [10] Photocapy / CC-BY-SA-2.0)

gangskontrolle vorhandenes Medium mitbenutzt.

Mautstationen

Schließlich eignen sich Smartcards auch zum Begleichen der Maut, die für die Passage diverser Schnellstraßen, Brücken oder Tunnel in unterschiedlichen Ländern erhoben wird. Gegenüber einer sonst für diese Zwecke verwendeten Vignette ergibt sich hiermit der Vorteil, dass die Abrechnung nach der tatsächlichen Nutzung erfolgen kann, statt eine Pauschale für einen bestimmten Zeitraum zu verlangen [2, S.827ff].

2.1.2 Authentifizierung

Mit der Authentifizierung wird üblicherweise die Zugangsberechtigung entweder zu einem realen Ort oder zu einem technischen System verbunden. Diese Authentifizierung ist nicht zwingend personengebunden, kann also auch anonym oder pseudonym (das heißt zum Beispiel unter Verwendung einer eindeutigen Identifikationsnummer) erfolgen. Oftmals werden hierbei kryptografische Schlüssel auf dem Chip gespeichert, um einerseits die Authentifizierung selbst abzusichern oder andererseits nachgelagerte Kommunikation zu schützen.

Gebäude

Um den Zugang zu Gebäuden wie Büros oder Parkhäusern, aber beispielsweise auch Sportstadien zu sichern, kann die Zugangsberechtigung auf einer Smartcard gespeichert werden. Im Umfeld von Unternehmen lässt diese sich gut in einen Mitarbeiterausweis oder ein Zugangstoken integrieren. Für Sportstadien und andere öffentliche Veranstaltungen kann der Smartcard-Chip im Ticket (Abbildung 2.3) untergebracht werden, um so eine schnelle automatische Einlasskontrolle zu ermöglichen.

Abbildung 2.4: Oyster-Karte (basierend auf [13] Frank Murmann / CC-BY-3.0)

Öffentlicher Personenverkehr

Genauso wie für Konzerte oder Fußballspiele können Tickets auch für den öffentlichen Personenverkehr ausgegeben werden, die dem Verwender das einmalige oder mehrmalige Befahren einer bestimmten Strecke ermöglichen. Alternativ ist es auch möglich, Start- und Endpunkt jeder Reise genau zu erfassen, um das entsprechende Entgelt zu berechnen und von einem Prepaid-Guthaben abzuziehen oder nachträglich in Rechnung zu stellen.

Die beispielhaft abgebildete Oyster-Card (Abbildung 2.4) kommt im Nahverkehr in London zum Einsatz. Sie kann sowohl als Guthabenkarte für einzelne Fahrten, als auch als Zeitkarte für beliebig viele Fahrten innerhalb eines bestimmten Zeitraums verwendet werden [11]. Ähnliche Angebote gibt es auch an vielen anderen Orten, etwa mit der Presto-Card in Kanada [12].

IT-Systeme

Zur Authentifizierung an IT-Systemen werden klassischerweise Passwörter verwendet, die jedoch unsicher sein können, wenn sie nicht bestimmten Komplexitätsvorgaben entsprechen oder von anderen Personen ausgespäht werden. Es bietet sich daher an, ein zusätzliches Authentifizierungsmedium in Form einer Smartcard einzusetzen, sodass neben Wissen (Passwort) auch Besitz (Smartcard) zur erfolgreichen Authentifizierung notwendig ist (Zwei-Faktor-Authentifizierung).

Smartcards können dazu auf verschiedene Weise mit dem System verbunden werden (Abbildung 2.5), sowohl als normale Karte über ein entsprechendes Lesegerät als auch direkt über einen integrierten USB-Anschluss. Zudem ist es möglich, von dem gespeicherten

Abbildung 2.5: Token (Aladdin [14] / CC-BY-SA-3.0)

Schlüsselmaterial einen sich in kurzen Zeitabständen ändernden Authentifizierungscode abzuleiten, der vom Benutzer manuell eingegeben werden muss, sodass keinerlei direkte Verbindung der Smartcard zum System erforderlich ist.

Der Einsatz einer Smartcard hat insbesondere den Vorteil, dass sie anders als ein Passwort nicht ohne weiteres dupliziert werden kann und so unbemerkt in den Besitz anderer Personen gelangt. Zudem lässt sich mit ihr leichter ein hohes Sicherheitsniveau erreichen, als mit einem Passwort entsprechender Komplexität, das sich ein Mensch merken muss.

Pay-TV

Werden Fernseh-Signale beispielsweise per Satellit übertragen, sind sie grundsätzlich für jeden Teilnehmer empfangbar. Soll ein Pay-TV-Angebot daher nur für zahlende Kunden zugänglich sein, muss der Datenstrom verschlüsselt werden, um alle anderen Teilnehmer auszuschließen. Um ein Vervielfältigen des Schlüsselmaterials eines Kunden zu vermeiden, kann dieses auf einer Smartcard gespeichert werden.

Diese lässt sich beispielsweise über ein so genanntes Conditional-Access-Module (Abbildung 2.6) mit dem Empfänger verbinden. Zusammen erzeugen sie den aktuell benötigten, jeweils nur kurze Zeit gültigen Schlüssel, um das TV-Signal zu entschlüsseln.

Mobilfunknetze

In Form von SIM-Karten (Subscriber Identity Module, Abbildung 2.7) werden Smartcards seit 1991 [16] millionenfach eingesetzt. Neben einer eindeutigen Identifikationsnummer

Abbildung 2.6: Conditional Access Module zur Aufnahme einer Smartcard in einen TV-Receiver (basierend auf [15] Arjan Almekinders / CC-BY-3.0)

(IMSI) speichern sie insbesondere einen zur Authentifizierung am Mobilfunknetz verwendeten Schlüssel, bieten darüber hinaus aber auch die Möglichkeit weitere Nutzdaten, etwa Telefonnummern oder Kurznachrichten, zu speichern. Oftmals werden die gespeicherten Daten über eine PIN gesichert, die vom Benutzer eingegeben werden muss, bevor der Zugriff gestattet wird [2, S.754].

2.1.3 Identifizierung

Bei der Verwendung von Smartcards zur Identifizierung steht die maschinenlesbare Speicherung und Übertragung von Identifizierungsmerkmalen, mitunter personenbezogenen Daten, im Vordergrund.

Ausweis

In Ausweise wie Personalausweise (Abbildung 2.8) oder Reisepässe sind Smartcards integriert, um, neben den in gewöhnlichen Ausweisen bereits in einer maschinenlesbaren Zone gespeicherten Daten, weitere Informationen, etwa eine digitale Kopie des Passbilds, unterzubringen [18]. Zusätzlich soll dies die Fälschungssicherheit erhöhen. Des Weiteren lassen sich mit einem solchen elektronischen Ausweis weitere Dienste verbinden, etwa die reine Altersbestätigung des Inhabers oder die Erstellung qualifizierter elektronischer Signaturen [19].

Krankenversichertenkarte

Seit 1995 dient die Krankenversichertenkarte als Nachweis des Versicherungsschutzes, wann immer ärztliche Leistungen in Anspruch genommen werden sollen. Sie hat den

Abbildung 2.7: SIM-Karte (Qurren [17] / CC-BY-SA-3.0)

Abbildung 2.8: Elektronischer Personalausweis der Bundesrepublik Deutschland [20]

Krankenschein aus Papier abgelöst, um den damit verbundenen Verwaltungsaufwand zu reduzieren. Auf ihr sind, in maschinenlesbarer Form, die zur Abrechnung der ärztlichen Behandlungen benötigten Daten, zum Beispiel Name und Krankenversichertennummer, gespeichert, wie sie auch auf dem Krankenschein vorhanden waren [2, S.822ff][21].

So wie die Krankenversichertenkarte den Krankenschein aus Papier abgelöst hat, sehen es Projekte wie die elektronische Gesundheitskarte zudem vor, weitere Papierdokumente durch digitale Daten zu ersetzen, etwa ärztliche Verordnungen als elektronisches Rezept zu speichern. Darüber hinaus ist es denkbar, medizinische Daten für die Notfallversorgung zu hinterlegen oder die ärztliche Dokumentation der Behandlung zu speichern (elektronische Patientenakte), um diese jederzeit verfügbar zu haben [22].

Elektronische Signatur

Um die Authentizität digitaler Daten sicherzustellen, können elektronische Signaturen verwendet werden, ähnlich wie Unterschriften auf Papierdokumenten. Die Signatur identifiziert dabei ihren Ersteller eindeutig und garantiert, dass die signierten Daten seit der Signaturerstellung unverändert sind. Die so genannte qualifizierte elektronische Signatur [23] kann dann auch genutzt werden, um die ansonsten gesetzlich vorgeschriebene schriftliche Form zu ersetzen. So lassen sich beispielsweise Verträge schließen, indem alle Parteien das Vertragsdokument elektronisch signieren, statt es handschriftlich zu unterschreiben [24].

Technisch werden dabei üblicherweise zwei Komponenten benötigt: ein geheimer Schlüssel, mit dem die Signatur über kryptografische Verfahren (beispielsweise RSA, dem bekanntesten Verfahren) erstellt wird, sowie ein Zertifikat, das den öffentlichen Schlüssel zur Überprüfung der Signatur an die Identität des Erstellers bindet und meist von einem vertrauenswürdigen Dritten, für qualifizierte elektronische Signaturen von einem gesetzlich definierten Zertifizierungsdiensteanbieter, erstellt wird. Um dieses Schlüsselmaterial sicher zu speichern, können wieder Smartcards, genannt Signaturkarten, verwendet werden [2, S.831ff][25].

Zeiterfassung

Vor mehr als hundert Jahren wurden Stechuhren erfunden, mit denen die Anwesenheitszeiten der Mitarbeiter erfasst werden, indem die aktuelle Uhrzeit beim Betreten und Verlassen des Betriebes auf eine Pappkarte gedruckt wird [26, S.5f]. Auch hier ermöglichte der technologische Fortschritt schon 1973 den Umstieg auf eine elektronische Zeiterfassung. Diese bietet nicht nur den Vorteil, die erfassten Daten automatisiert auswerten zu können, sondern lässt sich auch komfortabel mit anderen bereits vorgestellten Anwendungen, etwa der Zutrittskontrolle, im Mitarbeiterausweis zusammenfassen [26, S.13f].

Güter

Eine gänzlich andere Anwendung der Smartcard-Technologien ergibt sich bei der Identifikation von Gütern und anderen Gegenständen. Schon mit der Vorgängertechnologie,

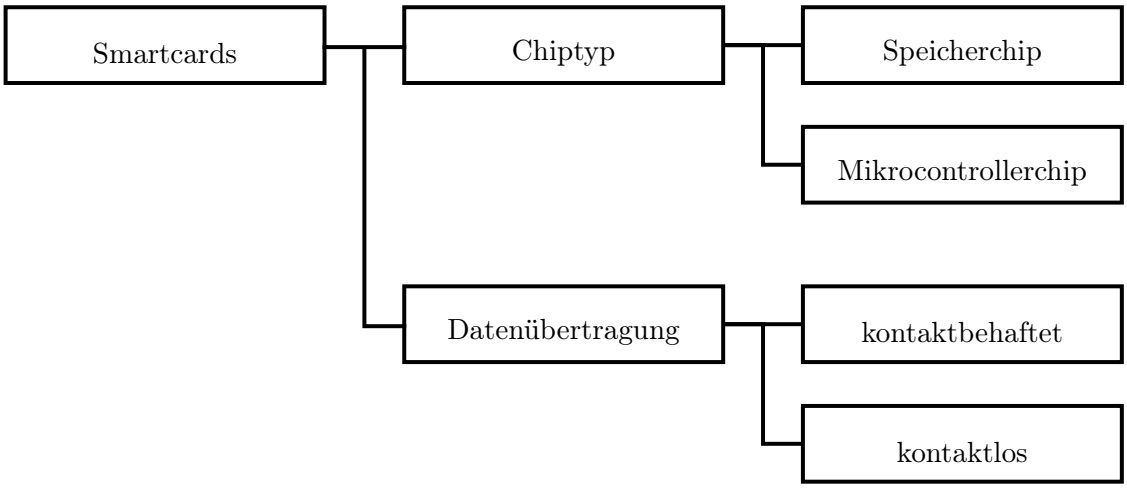

Abbildung 2.9: Klassifikationsmöglichkeiten von Smartcards [2, S.21]

den Barcodes, lassen sich eindeutige Identifikationsnummern an Gegenständen anbringen, über die etwa automatisierte Förderanlagen Pakete oder Gepäckstücke samt deren Zielorten erkennen können. Kontaktlose Smartcards (RFID) bieten neben einer höheren Speicherkapazität für komplexere Einsatzszenarien den Vorteil, dass zum Auslesen kein Sichtkontakt mehr bestehen muss und somit Gegenstände schneller und praktisch zeitgleich identifiziert werden können.

2.2 Funktionsweise

Der Aufbau und die Funktionsweise von Smartcards kann in verschiedenen Bereichen variieren (Abbildung 2.9). Zum einen kann es sich beim verwendeten Chip um einen einfachen Speicherchip oder einen komplexeren Mikrocontroller handeln. Zum anderen kann die Kommunikation mit der Außenwelt entweder kontaktbehaftet oder kontaktlos oder wahlweise über beide Schnittstellen (Dual-Interface) erfolgen [2, S.20ff].

2.2.1 Chiptyp

Je nach Anwendung können unterschiedliche Arten von Chips verwendet werden, die sich hinsichtlich ihrer Leistungsfähigkeit, aber auch des Fertigungsaufwands und damit der Kosten unterscheiden.

Speicherchip

Speicherkarten sind der einfachere und günstigere Vertreter der Chipkarten. Sie bestehen meist aus wenig mehr als einem von außen adressierbaren Speicher, optional geschützt durch eine Sicherheitslogik, die Schreib- oder Lesezugriffe auf einzelne Speicherbereiche nicht oder nur nach vorhergehender Authentifizierung zulässt. Oftmals werden die Chips

auf die beabsichtigte Anwendung, etwa den Einsatz als Telefonkarte, zugeschnitten, was zusammen mit dem einfachen Aufbau eine sehr preisgünstige Fertigung ermöglicht [2, S.21f].

Mikrocontrollerchip

Im Gegensatz zu einer Speicherkarte beinhaltet eine Mikroprozessorkarte einen Prozessor, der beliebigen Programmcode ausführen kann und so einen vergleichsweise flexiblen Einsatz ermöglicht. Der Programmcode - das Betriebssystem des Chips - wird während der Herstellung in einem unveränderlichen Speicher (Read-Only Memory, kurz ROM) abgelegt. Zusätzlich dient der RAM (Random-Access Memory) als Arbeitsspeicher des Prozessors während der Ausführung. Daten können, wie auch bei der Speicherkarte, etwa in einem Electrically Erasable Programmable Read-Only Memory (EEPROM) dauerhaft gespeichert werden [2, S.22f].

2.2.2 Datenübertragung

Die Datenübertragung zwischen Karte und Lesegerät kann sowohl kontaktbehaftet als auch kontaktlos erfolgen. Dabei ist es auch möglich, beide Schnittstellen in einer Karte zu vereinen, entweder indem ein Chip mit beiden Schnittstellen verbunden wird (Dual-Interface) oder indem zwei getrennte Chips, einer je Schnittstelle, in die Karte eingebracht werden.

Die Kommunikation wird dabei in allen Fällen vom Leser initiiert, die Karte reagiert nur auf dessen Befehle. Dieses Master-Slave-Verhalten vereinfacht die Kommunikationsprotokolle [2, S.377].

Kontaktbehaftet

Kontaktbehaftete Chipkarten sind die ältere der beiden Formen. Sie kommen üblicherweise in den im ISO-Standard 7810 festgelegten Formaten, beispielsweise dem weit verbreiteten Scheckkarten-Format (ID-1), daher [2, S.30]. Im ISO-Standard 7816 werden ergänzend weitere Eigenschaften definiert, unter anderem die Lage und Beschaltung der Kontakte sowie Kommunikationsprotokolle.

Von den acht definierten Kontakten (Abbildung 2.10) sind zwei normalerweise unbenutzt (C4 und C8), ein weiterer wird nicht mehr verwendet (C6). Die übrigen Kontakte dienen zur Stromversorgung des Chips (C1 und C5), als Eingang für Takt (C3) und Reset-Signal (C2) sowie zur Kommunikation (C7) [2, S.56].

Unterhalb der Kontakte befindet sich der eigentliche Chip (Abbildung 2.11), der oftmals über dünne Drähte mit den Goldkontakten verbunden wird, bevor diese zusammen in die Plastikkarte eingeklebt werden [2, S.45ff].

Die Kontakte einer Chipkarte sind auch ihr größter Nachteil, da sie eine der häufigsten Fehlerquellen sind. So können Verschmutzungen oder Abnutzungen zu Störungen führen, oder Vibrationen zu kurzzeitigen Kontaktunterbrechungen. Elektrostatische Entladungen, die über die freiliegenden Kontakte direkt an den Chip gelangen, können dessen Funktionsfähigkeit gefährden [2, S.23]. Zudem muss beim Einführen der Karte in ein Lesegerät

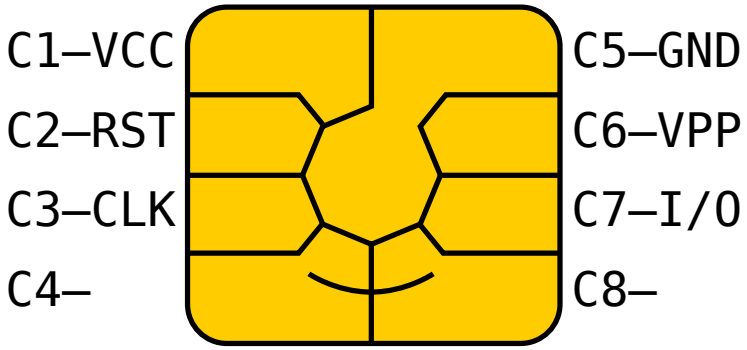

Abbildung 2.10: Kontakte einer Chipkarte (Dacs, WhiteTimberwolf [27] / CC-BY-SA-3.0)

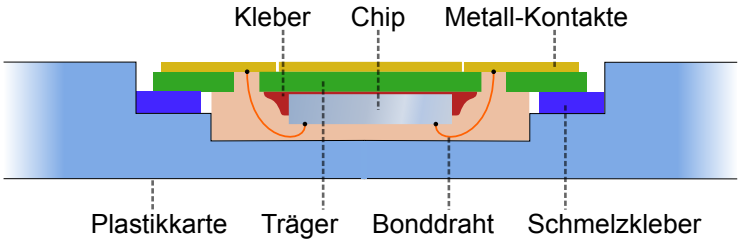

Abbildung 2.11: Aufbau einer kontaktbehafteten Chipkarte (Querschnitt) (basierend auf [28] Justin Ormont / CC-BY-SA-3.0)

stets die korrekte Orientierung beachtet werden, damit ein Kontakt überhaupt zustande kommen kann.

Kontaktlos

Viele der beschriebenen Probleme einer kontaktbehafteten Karte werden mit der kontaktlosen Variante vermieden. Zudem ergeben sich neue Anwendungsmöglichkeiten, da nicht mehr zwingend ein direkter Kontakt zwischen Karte und Lesegerät erforderlich ist. Vielmehr ist es denkbar, dass Karten über eine Entfernung von bis zu einem Meter zum Beispiel im Vorbeigehen ausgelesen werden und Türen automatisch öffnen [2, S.23f]. Auch die Formate, wie sie in ISO 7810 festgelegt werden, müssen nicht mehr unbedingt eingehalten werden, sodass sich kontaktlose Chips in unterschiedliche Gegenstände integrieren lassen.

Was bei kontaktbehafteten Karten über elektrische Kontaktflächen realisiert wird - Stromversorgung und Datenübertragung - muss bei kontaktlosen Karten über die Funkschnittstelle ermöglicht werden. Zur Energieübertragung wird nach dem Prinzip des lose gekoppelten Transformators vom Lesegerät ein hochfrequentes elektromagnetisches Feld erzeugt, das in eine Spule in der Karte eine Spannung induziert [2, S.96]. Die Datenübertragung von Lesegerät zu Karte geschieht meist mittels Amplituden- oder Phasenmodulation, in entgegengesetzte Richtung mit der so genannten Lastmodulation, bei der die Karte je nach zu übertragender Information mehr oder weniger Energie vom Lesegerät bezieht [2, S.97].

Wie auch für kontaktbehaftete Karten werden diese und viele weitere Eigenschaften in mehreren Standards festgelegt, neben ISO 10536 (Close-coupled Cards) und ISO 15693 (Vicinity Cards) ist ISO 14443 (Proximity Cards) am gebräuchlichsten. Letzterer beschreibt Karten im 13,56 MHz-Bereich mit einer Ausleseentfernung von etwa zehn Zentimetern [2, S.107f].

Ein Problem beim Einsatz kontaktloser Karten ergibt sich, sobald mehrere Karten gleichzeitig in Reichweite eines Lesegeräts sind. Um Störungen der Kommunikation zu vermeiden, darf immer nur eine Karte gleichzeitig aktiv sein. Daher benötigt das Lesegerät eine Möglichkeit, alle vorhandenen Karten zu ermitteln und gezielt einzelne auszuwählen. Dies geschieht mit einem so genannten Antikollisionsverfahren [2, S.99].

Gemäß ISO 14443-3 (Typ A) funktioniert dies wie folgt: Jede Karte wird über eine eindeutige Nummer (UID) identifiziert. Diese UID sendet jede Karte nach Aufforderung durch das Lesegerät bitweise, sodass Kollisionen einzelner Bits erkannt werden können. Somit kann das Lesegerät das größtmögliche UID-Präfix ermitteln, das für alle Karten gleich ist (alle Bits bis zur ersten Kollision) und dies um ein 0- oder 1-Bit ergänzen. Anschließend ergeht die Aufforderung an alle Karten, deren UID zu dem gebildeten Präfix passt, erneut die noch fehlenden UID-Bits zu übermitteln. So setzt sich das Verfahren fort, bis die vollständige UID ohne Kollisionen ermittelt wurde. Zum Auflisten aller Karten muss es bei jeder Bit-Kollision für beide möglichen Werte wiederholt werden [2, S.113ff].

2.3 Sicherheit

Anforderungen an Smartcard-Anwendungen lassen sich in zwei Kategorien einteilen: Funktionalität und Sicherheit. Während die korrekte Funktionalität von jedem Kunden einfach überprüft werden kann („Öffnet die Tür, wenn ein gültiger Ausweis präsentiert wird?“), ist die vollständige Überprüfung aller Sicherheitsanforderungen („Kann ein Ausweis mit vertretbarem Aufwand geklont werden?“) ungleich schwerer, da auch nur der kleinste Fehler an irgendeiner Stelle die Sicherheit des Gesamtsystems untergraben kann [29].

Dies führte in der Vergangenheit zu erfolgreichen Angriffen auch auf weit verbreitete Systeme, deren Ansatzpunkte sowie mögliche Präventivmaßnahmen seitens der Entwickler und Anwender im Folgenden näher erläutert werden.

2.3.1 Angriffstechniken

Viele Angriffe richten sich direkt gegen die Hardware und die damit realisierten Protokolle. Angriffe, die auf den Nutzer selbst abzielen (Social Engineering), werden hier nicht berücksichtigt.

Protokollanalyse

Der einfachste Angriff besteht in der Analyse der übertragenen Daten, um Rückschlüsse auf das verwendete Protokoll zu ziehen und mögliche Schwachstellen auszunutzen. Kommt keine Verschlüsselung zum Einsatz, können die Daten im Klartext mitgelesen werden. Ohne Authentifizierung können Karte oder Leser imitiert werden. Schwache Algorithmen oder Fehler im Protokolldesign ermöglichen es mitunter, diese Schutzmaßnahmen zu brechen oder zu umgehen [30, S.9].

Relay

Kontaktlose Smartcards sind aufgrund der verwendeten Sendeleistung normalerweise nur im Abstand von einigen Zentimetern bis etwa einem Meter verwendbar, sodass der Inhaber seine Karte aktiv in die Nähe des Lesegeräts bringen muss, um zum Beispiel eine Tür zu öffnen. Ein Relay-Angriff versucht diese Beschränkung zu umgehen, indem eigene Geräte sowohl mit Karte als auch mit Leser kommunizieren und deren Daten unverändert über einen anderen Kanal, etwa eine leistungsfähigere Funkverbindung, an das jeweils andere Ende weiterleiten.

Angriffe dieser Art funktionieren unabhängig von verwendeten Verschlüsselungsmethoden, da die ausgetauschten Daten auf Protokollebene unverändert sind. Das angegriffene System müsste beispielsweise die Antwortzeiten auswerten, um festzustellen, dass sich sein Kommunikationspartner in größerer Entfernung befindet [30, S.7].

Seitenkanal-Attacken

Sind die verwendeten Algorithmen aus kryptografischer Sicht nicht angreifbar, können Seitenkanal-Attacken auf die konkrete Implementierung dennoch helfen, geheime Daten zu ermitteln. Dabei macht man es sich zu nutze, dass, ohne entsprechende Gegenmaßnahmen, das Verhalten des Chips Rückschlüsse auf den ausgeführten Befehl und die verarbeiteten Daten zulässt. Dazu misst man beispielsweise den Stromverbrauch [2, S.548ff] oder die Dauer bestimmter Operationen [2, S.561] oder verursacht gezielt Fehler bei der Berechnung [2, S.562f]. Meist sind dabei viele Wiederholungen der Messungen notwendig, die anschließend mit statistischen Methoden ausgewertet werden.

Hardware Reverse Engineering

Die oft aufwendigsten Angriffe bestehen im Reverse Engineering der Hardware von Karte oder Leser. Dabei kann es sowohl Ziel sein, die Funktionsweise der verwendeten Algorithmen zu verstehen, als auch in Karte oder Leser gespeicherte geheime Informationen, zum Beispiel Schlüssel, zu extrahieren. Die Analyse der Chips kann dabei einerseits im abgeschalteten Zustand erfolgen, wobei es mit entsprechender Ausstattung möglich ist, die komplette Schaltung zu rekonstruieren. Andererseits kann im laufenden Betrieb versucht werden, Daten auf Busleitungen oder im Speicher abzugreifen [2, S.535ff].

2.3.2 Bekannte Angriffe

Im Folgenden werden einige Angriffe auf bekannte Systeme beschrieben, deren Ziel es ist, eine Karte beliebig zu manipulieren oder zu imitieren, indem sowohl das Kommunikationsprotokoll als auch das auf einer Karte gespeicherte, geheime Schlüsselmaterial in Erfahrung gebracht wird.

Mifare Classic

Als eine der ersten weit verbreiteten kontaktlosen Smartcard-Lösungen wurden für Mifare Classic Ende 2007 Angriffsmöglichkeiten öffentlich bekannt [31]. Bei diesem Produkt kommt der proprietäre Verschlüsselungsalgorithmus Crypto-1 zum Einsatz, für den bis zu diesem Punkt keine Informationen, nicht einmal eine Software-Implementierung, veröffentlicht wurden. Die einzige Möglichkeit, an den Algorithmus zu gelangen, war es daher, dessen Hardware-Implementierung in den Chips selbst zu untersuchen. Dazu wurden diese aus den Plastikkarten gelöst, so weit poliert, dass die einzelnen Schichten sichtbar wurden, und diese unter einem Mikroskop fotografiert [32, S.1ff].

Aus diesen Bildern konnten in einem größtenteils automatisierten Vorgang die einzelnen Gatter und deren Verbindungen rekonstruiert werden, sodass schließlich die Funktionsweise der gesamten Schaltung bekannt war. Weitere Informationen, etwa die benötigten Eingaben, wurden aus einer Analyse der Protokoll-Kommunikation gewonnen [32, S.3f].

So konnten schließlich verschiedene Schwachstellen gefunden werden. Der verwendete Pseudo-Zufallszahlengenerator, realisiert als linear rückgekoppeltes Schieberegister (LFSR), wird, sobald die Stromversorgung der Karte hergestellt ist, mit einem konstanten

Wert initialisiert und erzeugt fortan eine immer gleiche Folge von Zufallszahlen, die sich nach gut einer halben Sekunde wiederholt [32, S.5f]. Ein Angreifer muss daher nur das Timing einer aufgezeichneten Kommunikation exakt nachstellen (Replay-Angriff), um sich gegenüber der Karte als legitimes Lesegerät auszugeben und eine zuvor durchgeführte Aktion, etwa „erhöhe das gespeicherte Guthaben“, zu wiederholen [33, S.62f].

Auch die Verschlüsselung Crypto-1 ist angreifbar. Die verwendete Schlüssellänge beträgt nur 48 Bit und ermöglicht damit eine vollständige Schlüsselsuche (Brute-Force-Angriff), je nach Implementierung, in unter einer Stunde [32, S.5]. Des Weiteren existieren algebraische Angriffe, die mit der Aufzeichnung einer einzigen Transaktion den verwendeten Schlüssel berechnen können, indem die Verschlüsselung als großes Gleichungssystem dargestellt wird, welches in wenigen Minuten von einem handelsüblichen Computer gelöst werden kann [34, S.2][33, S.66]. Schließlich wurden auch Angriffe entwickelt, die ganz ohne aufgezeichnete Kommunikation nur mit einer ausreichenden Anzahl Abfragen an die anzugreifende Karte auskommen (Card-Only-Angriff) [35, S.6f].

Legic Prime

Wie schon bei Mifare Classic war auch bei Legic Prime die eingesetzte Verschlüsselung (Legic RF) zunächst nicht bekannt. Statt hier jedoch den Weg über die Analyse der Hardware zu gehen, ließ sich das Verfahren einzig durch Beobachtung und Manipulation der Funkkommunikation zwischen Leser und Karte offenlegen. Dabei zeigte sich unter anderem, dass mangels geheimer Schlüssel gar nicht von einer Verschlüsselung der Kommunikation in diesem Sinne gesprochen werden kann, sondern nur eine relativ aufwendige Verschleierung stattfindet. Dies erlaubt es problemlos, sämtliche in Umlauf befindlichen Legic-Prime-Karten zu manipulieren oder gänzlich neue Karten zu erstellen. Weitere Details dieses Systems und der Angriffe werden in Abschnitt 3 beschrieben.

Mifare DESFire

Im Gegensatz zu Mifare Classic und Legic Prime wurde bei Mifare DESFire (MF3 IC D40) bereits auf bekannte und als sicher geltende kryptografische Verfahren wie DES gesetzt, was ähnliche Angriffe zunächst ausschließt. Stattdessen bietet sich die Möglichkeit, den Chip während der Laufzeit der Algorithmen zu beobachten und aus den so gewonnenen Seitenkanal-Informationen auf den internen Zustand und damit den verwendeten Schlüssel zu schließen. Die Auswertung des Stromverbrauchs, der sich je nach ausgeführter Operation oder verarbeiteten Daten ändern kann, ermöglicht es, den geheimen Schlüssel innerhalb weniger Stunden zu ermitteln [36, S.11]. Der Nachfolger DESFire EV1 ist gegen diesen Angriff jedoch resistent [36, S.15].

2.3.3 Abwehrmaßnahmen

Genau wie Angriffe auf verschiedenen Ebenen ansetzen können, lassen sich auch Abwehrmaßnahmen danach einteilen. Auf der physikalischen Ebene wird das Design der Hardware so optimiert, dass ein Angreifer möglichst viel Aufwand betreiben muss, um nützliche Informationen zu gewinnen. Sicherheit auf der logischen Ebene wird durch den Einsatz

sicherer Algorithmen und Protokolle sowie deren korrekter Implementierung angestrebt. Organisatorische Maßnahmen dienen schließlich dazu, trotz aller Bemühungen erfolgreiche Angriffe auf einzelne Komponenten zu erkennen und deren Effekte abzuschwächen.

Physikalisch

Um eine Rekonstruktion der Hardware zu erschweren, kann das Design der Chips möglichst komplex gestaltet werden, indem darauf verzichtet wird, in ihrer Funktionsweise bekannte Standardzellen einzusetzen, oder indem an sich funktionslose Dummy-Strukturen zusätzlich integriert werden. Eine Analyse im laufenden Betrieb lässt sich durch das verdeckte Verbauen von Bussen und Speicher in den unteren, nicht ohne weiteres zugänglichen Schichten des Chips behindern [2, S.538f].

Zusätzlich lassen sich die Inhalte von Bussen und Speichern bereits auf Hardwareebene durch Scrambling, das heißt das Vertauschen von Adressen, oder durch Verschlüsselung der Daten schützen [2, S.540f][2, S.544f]. Schutzschichten, die den gesamten Chip umgeben und von diesem permanent auf ihre Unversehrtheit geprüft werden, bilden eine weitere Barriere die interne Funktionsweise zu analysieren [2, S.539f]. Andere Sensoren, etwa zur Überwachung von Spannung oder Frequenz, können den Chip abschalten, sobald diese einen Betrieb außerhalb der zulässigen Spezifikationen und damit einen möglichen Angriff erkennen [2, S.542ff].

Seitenkanal-Angriffe auf Basis des Stromverbrauchs lassen sich unter anderem dadurch vermeiden, dass durch entsprechende Hardware-Schaltungen ein konstanter (aber auch höchstmöglicher) Stromverbrauch sichergestellt wird. Ebenso ist es denkbar, zeitgleich zu kritischen Operationen dabei nicht benötigte Komponenten zu aktivieren und Zufallsdaten verarbeiten zu lassen, um so ein künstliches Rauschen des Stromverbrauchs zu generieren. Auch zufällige Wartezeiten während der Verarbeitung können die Analyse stören [2, S.551].

Logisch

Grundvoraussetzung für ein sicheres System ist eine Authentifizierung der jeweiligen Gegenstelle bei der Kommunikation, da nur so sicher gestellt werden kann, dass es sich beim Kommunikationspartner nicht um einen Angreifer handelt [2, S.571]. Obwohl es auch möglich ist, allein die Integrität übertragener Daten zu schützen, werden diese oftmals zusätzlich verschlüsselt, sodass auch die Vertraulichkeit gewährleistet ist [2, S.570]. Alle für derartige kryptografische Operationen eingesetzten Algorithmen sollten öffentlich bekannt und untersucht sein, um Sicherheitslücken soweit wie möglich auszuschließen [30, S.9].

Bei der Implementierung der Algorithmen ist darauf zu achten, dass ein konstantes Zeitverhalten gewährleistet wird, das heißt unabhängig von den Eingabedaten die gleiche Ausführungszeit benötigt wird, um so darauf basierende Seitenkanalattacken zu unterbinden [2, S.560f]. Zufallszahlen, die unter anderem bei der Authentifizierung eine wichtige Rolle spielen, sind auf kryptografisch sichere Weise zu erzeugen [2, S.567f][30, S.9].

Organisatorisch

Auf organisatorischer Ebene ist es wichtig, die Verwendung von kartenindividuellem Schlüsselmaterial sicherzustellen, um zu verhindern, dass ein erfolgreicher Angriff auf eine Karte auch andere Karten unmittelbar gefährdet. Dies geschieht entweder durch den Einsatz asymmetrischer Kryptografie, bei der für jede Karte eigenes Schlüsselmaterial generiert und von einer zentralen Instanz signiert wird, oder durch die so genannte Key Diversification bei Verwendung symmetrischer Kryptografie. Hierbei wird unter Einsatz einer Einwegfunktion ein kartenindividueller Schlüssel von einem Hauptschlüssel abgeleitet und nur der abgeleitete Schlüssel auf der Karte gespeichert [30, S.22].

Sollte es zu einem Verlust oder einem erfolgreichen Angriff auf eine Karte kommen, muss es möglich sein, diese Karte (genauer: ihr Schlüsselmaterial) im System zu sperren, indem entsprechende Sperrlisten gepflegt werden [2, S.572]. Dass eine Karte angegriffen und beispielsweise geklont wurde, lässt sich durch das Erstellen von Logeinträgen für jede Verwendung und deren Überprüfung auf Unregelmäßigkeiten erkennen [30, S.10f][30, S.23f].

Als zusätzlicher Schutz für besonders gefährdete Bereiche lässt sich die so genannte Multi-Faktor-Authentifizierung einsetzen, bei der der Benutzer neben dem Besitz (der Smartcard) auch noch Wissen (beispielsweise ein Passwort) oder ein biometrisches Merkmal (etwa einen Fingerabdruck) vorweisen muss. Selbst wenn eins dieser Authentifizierungsmerkmale in die Hände eines Angreifers fällt, verhindern die anderen nach wie vor den Zugriff [30, S.11f].

3 Legic Prime

Legic Prime wurde 1992 von der schweizerischen Legic Identsystems AG [37] als Zutrittskontrollsystem im Unternehmens- und Freizeitbereich vorgestellt [38]. Obwohl es heute hauptsächlich zur Vereinfachung von Organisationsprozessen und zur Erhöhung des Komforts vermarktet wird, kam Legic Prime auch in potentiell sicherheitskritischen Umgebungen zum Einsatz, etwa in Kernkraftwerken [39] und Flughäfen [40].

3.1 Funktionsweise

Bei Legic Prime handelt sich um ein kontaktloses System, dessen Chips sich in Karten, aber auch Schlüsselanhänger oder Uhren integrieren lassen. Diese können dann über eine Distanz von bis zu siebzig Zentimetern noch ausgelesen werden. Es stehen verschiedene Speichergrößen zur Verfügung, am gebräuchlichsten sind 256 Byte (MIM256) und 1024 Byte (MIM1024) [41]. Ein MIM22 bietet mit 22 Byte gerade genug Platz für ein Master-Token (Abschnitt 3.1.2) und wird ausschließlich dafür verwendet [42, S.6].

3.1.1 Segmente

Legic Prime unterstützt die Verwendung einer Karte für mehrere verschiedene Anwendungen, beispielsweise neben der Zutrittskontrolle auch eine Bezahlfunktion. Dazu ist es möglich, eine Karte in mehrere unabhängige Segmente (maximal 127 [43, S.10]) zu unterteilen, die den einzelnen Anwendungen zugeordnet werden [42, S.11].

Nicht segmentierte Medien sind ebenfalls möglich, kommen jedoch ausschließlich noch bei Master-Token (Abschnitt 3.1.2) zum Einsatz [44][43, S.9].

3.1.2 Master-Token System Control

Ein als Alleinstellungsmerkmal beworbenes Charakteristikum der Legic-Lösungen ist eine Master-Token System Control (MTSC) genannte Technologie [45]. Grundidee dabei ist es, Authentifizierungsinformationen in Form von physischen Tokens zu speichern statt beispielsweise Passwörter zu verwenden. Diese Verwendung von Besitz statt Wissen zur Authentifizierung soll eine bessere Kontrolle ermöglichen, da ein Token zum Beispiel sicher in einem Safe verwahrt und nicht unbemerkt weitergegeben oder vervielfältigt werden kann.

Legic Prime nutzt verschiedene Arten von Token [44], die einerseits benötigt werden, um neue Karten zu erstellen, und andererseits Lesegeräte für den Umgang mit bestimmten Karten zu autorisieren. Diese Token sind dabei im Grunde normale Legic-Prime-Medien, lediglich einige Bit im Speicher kodieren die Spezialfunktionen [43, S.13].

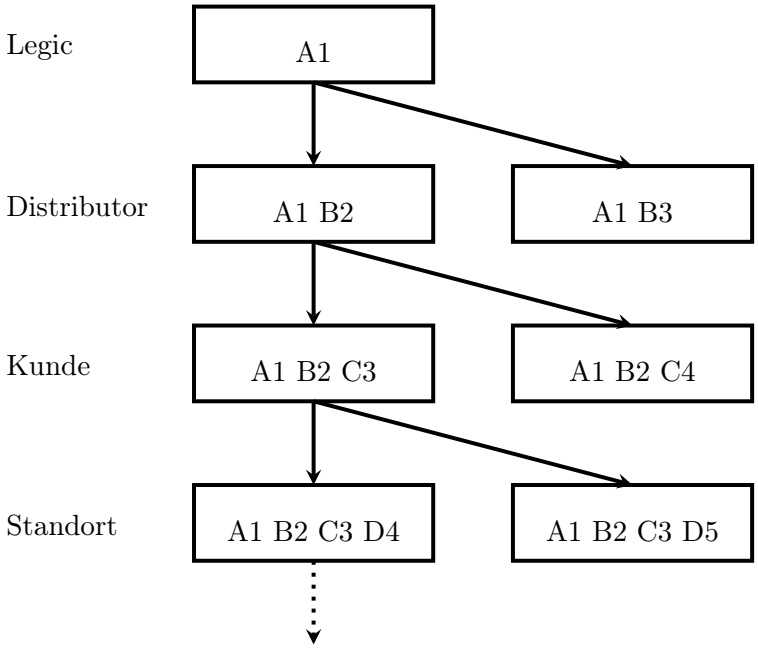

Abbildung 3.1: Hierarchie der Token mittels MTSC

Alle Karten (genauer: Segmente auf einer Karte [43, S.10]) und Lesegeräte enthalten dabei einen von Legic so genannten genetischen Code [45, S.3], implementiert als eine Folge von Bytes. Eine Lesegerät ist dabei nur dann für eine Karte autorisiert, wenn der eigene Code Präfix des Kartencodes ist. Gleichzeitig können Master-Token Sub-Token erzeugen, indem der eigene genetische Code um mindestens ein Byte erweitert wird. So entsteht eine Hierarchie von Berechtigungen (Abbildung 3.1), angelehnt an konventionelle Schließsysteme, bei denen manche Schlüssel nur einzelne Türen öffnen, während andere als Generalschlüssel für viele Türen verwendet werden können. Ebenso lassen sich damit Organisationhierarchien abbilden, sodass beispielsweise ein Unternehmen Sub-Token für seine verschiedenen Standorte erzeugen kann, die wiederum nach Abteilungen aufgeteilt werden können. Zu beachten ist dabei jedoch, dass an der Wurzel immer Legic oder einer ihrer Lizenznehmer steht [42, S.9f].

Die General Autorisation Media (GAM) ermöglichen es, Sub-Tokens zu erstellen. Ein Identification Autorisation Media (IAM) beziehungsweise Extended Autorisation Media (XAM) dienen der temporären beziehungsweise dauerhaften Berechtigung der Legic Medium-Kodierstation, um zuvor definierte Segmente auf Karten zu erstellen. Die System Autorisation Media (SAM) sind das Gegenstück, mit denen Lesegeräte die Berechtigung erhalten, vorhandene Segmente auf Karten zu beschreiben. Ein SAM63, genannt Taufkarte, berechtigt den Leser, während ein SAM64, genannt Enttaufkarte, die Berechtigung wieder entzieht. Ein SAM4, genannt Parametrierkarte, dient zur allgemeinen Konfiguration der Lesegeräte [44].

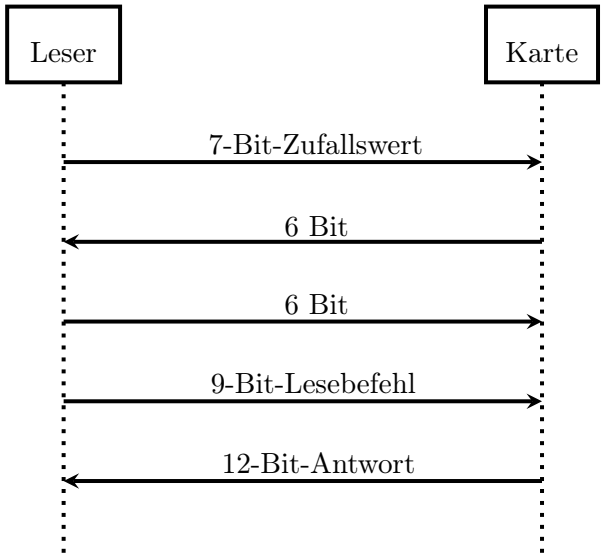

Abbildung 3.2: Legic-Prime-Kommunikationsprotokoll

3.1.3 Kommunikationsprotokoll

Legic Prime benutzt zwar die auch heute für RFID-Anwendungen verwendete Funkfrequenz von 13,56 MHz, basiert jedoch auf keinem der dafür gebräuchlichen ISO-Standards 14443 bzw. 15693, da diese zum Zeitpunkt der Entwicklung noch nicht existierten. Es wurde von Seiten von Legic jedoch versucht, ihr System als Anhang F des ISO-Standards 14443 einzubringen, was allerdings abgelehnt wurde [43, S.4]. Die in diesem Prozess veröffentlichten Unterlagen sind jedoch eine der wenigen öffentlichen Quellen, die die Funktionsweise, in diesem Fall die Übertragung einzelner Bits, genauer beschreiben [43, S.8]. Alle anderen Informationen stammen aus der detaillierten Untersuchung von Henryk Plötz und Karsten Nohl, die ihre Ergebnisse erstmals Ende 2009 auf dem 26. Chaos Communication Congress vorstellten [46].

Zu Beginn jeder Sitzung (Abbildung 3.2) übermittelt der Leser einen 7-Bit-Zufallswert an die Karte, der als Initialisierungsvektor einer Stromchiffre, realisiert durch eine Kombination aus zwei linear rückgekoppelten Schieberegistern (LFSR), verwendet wird. Alle folgenden Nachrichten werden mit der Ausgabe des Generators per XOR verknüpft [43, S.8]. Die Karte sendet anschließend sechs Bit, die ihren Typ kodieren, was mit einer weiteren 6-Bit-Nachricht des Lesers bestätigt wird. Damit ist die Initialisierung abgeschlossen [43, S.9].

Alle weitere Kommunikation kann aus genau zwei Befehlen bestehen, dem Lesen von einer Adresse der Karte und dem Schreiben eines Werts an eine bestimmte Adresse der Karte. Bei jeder Übertragung von Nutzdaten, das heißt für die Antwort der Karte auf einen Lesebefehl und für einen Schreibbefehl, wird zusätzlich eine Prüfsumme (CRC-4) angehängt [43, S.9].

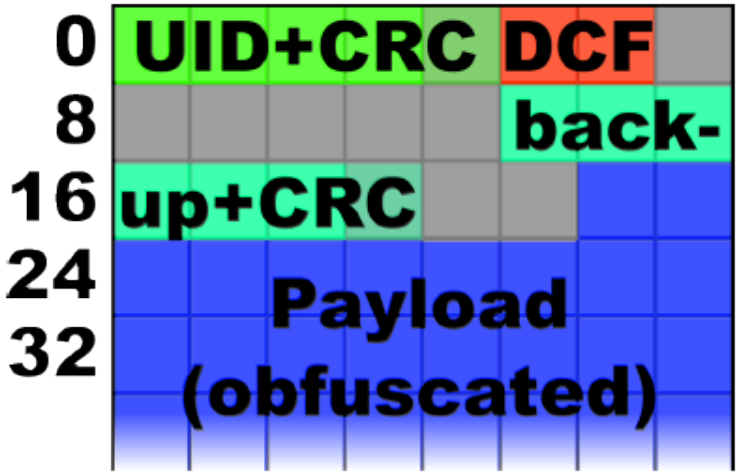

Abbildung 3.3: Speicherstruktur einer Legic-Prime-Karte [42, S.49]

3.2 Sicherheitsprobleme

Verschiedene Schwachstellen ermöglichen es einerseits, beliebige Karten vollständig zu emulieren, als auch andererseits, einen eigenen Leser zu kreieren, der fast ohne Beschränkungen Karten auslesen und beschreiben kann. So können problemlos neue Karten erstellt sowie vorhandene geklont werden, womit Legic Prime praktisch keinerlei Sicherheit bietet.

3.2.1 Fehlende Hardwaresicherheit

Wie die Analyse des Protokolls gezeigt hat, werden auf Seiten der Hardware kaum Beschränkungen umgesetzt, sodass mit einer eigenen Implementierung eines Lesegeräts praktisch beliebige Lese- und Schreibzugriffe auf eine Karte möglich sind, auch wenn Bereiche eigentlich als schreibgeschützt oder unlesbar gekennzeichnet sind. Nur Legics eigene Leser befolgen derartige Anweisungen [43, S.11].

Von der Karte selbst werden nur zwei Beschränkungen umgesetzt, die damit auch für alle anderen Leser gelten. Zum einen gilt für die ersten fünf Byte (Abbildung 3.3, Eintrag UID), in denen die UID der Karte (vier Byte) sowie ein CRC über die UID (ein Byte) gespeichert sind, ein Schreibschutz. Zum anderen werden die nächsten beiden Byte zusammen als Integer betrachtet (Abbildung 3.3, DCF) und können nur mit kleineren Werten überschrieben werden [43, S.9]. Diese beiden Bytes werden zur Konfiguration des MTSC verwendet und stellen mit dieser Logik sicher, dass ein bestehendes Token nicht zu einem mächtigeren Token aufgewertet werden kann [43, S.13f]. Bei leeren, fabrikneuen Karten sind die Bytes allerdings auf den maximalen Wert gesetzt, sodass diese Karten als beliebige Tokens programmiert werden können.

3.2.2 Schwache Verschlüsselung

Auch die eingesetzte Verschlüsselung der Kommunikation ist aus mehreren Gründen ungeeignet. Zunächst ist festzustellen, dass nur der Leser einen zufälligen Initialisierungsvektor beisteuert, womit die Karte ohne weiteres anfällig für Replay-Angriffe ist [43, S.5]. Auch der Zufallszahlengenerator des Lesers ist schwach. Zum einen sind die Werte zu klein, um Kollisionen in der Praxis wirksam zu vermeiden. Zum anderen wurde beobachtet, dass ein bestimmter Wert in etwa zehn Prozent der Fälle - und damit wesentlich häufiger als bei einer Gleichverteilung - auftritt [42, S.22].

Des Weiteren fehlen die zu einer Verschlüsselung notwendigen geheimen Schlüssel völlig, das heißt der verwendete Schlüsselstrom hängt einzig vom Initialisierungsvektor des Lesers ab, der zwangsweise im Klartext übertragen wird. Diese Verschlüsselung ist somit im Grunde nichts weiter als eine Verschleierung, die leicht umgangen werden kann [42, S.34].

3.2.3 Fragwürdige Verschleierung

Nutzt man einen eigenen Leser zum Zugriff auf die Karte zeigt sich, dass die Nutzdaten (Abbildung 3.3, Eintrag Payload) dort nicht im Klartext gespeichert werden, sondern deren Bytes per XOR mit dem CRC der UID verknüpft sind. Dieser ist aber frei lesbar auf der Karte gespeichert, sodass der Klartext leicht rekonstruiert werden kann und es sich dabei um nicht mehr handelt als eine zusätzliche Schicht der Verschleierung („Obscurity in Depth“ [42, S.48]) ohne kryptografische Funktion.

3.2.4 Unsichere Implementierung

Mindestens zwei Aspekte des Systems sind so implementiert, dass einem Angreifer mehr Informationen preisgegeben werden als unbedingt notwendig. Zum einen werden beim Löschen von Daten (Segmenten) diese nicht überschrieben, sondern lediglich als ungültig gekennzeichnet, sodass sie nach wie vor ausgelesen werden können. Zum anderen werden bei bestimmten Operationen Sicherheitskopien von Daten auf der Karte gespeichert (Abbildung 3.3, Eintrag backup), um den vorherigen Stand wiederherstellen zu können, sollte die Operation unterbrochen werden. Auch diese Kopien werden anschließend nur als ungültig markiert, ohne sie wirklich zu löschen, sodass sie Aufschluss über die durchgeführte Operation geben können [43, S.11].

Eine weitere Schwachstelle liegt im MTSC. Da auch Master-Tokens gewöhnliche Legic-Prime-Karten sind, können sie genauso wie diese geklont oder neu erzeugt werden. Doch selbst wenn dies nicht das Fall wäre, sind sie zwangsläufig klonbar, denn es genügt, diese einmalig einem Leser zu präsentieren, um diesem beispielsweise für die folgenden zehn Minuten zu erlauben, neue Karten zu erstellen. Es existiert nun also eine temporäre Kopie im Leser, da das Master-Token selbst anschließend nicht mehr anwesend sein muss. Erzeugt man etwa alle fünf Minuten eine neue Karte und setzt dadurch den Timeout zurück, lässt sich die Gültigkeit sogar beliebig verlängern [47][42, S.52].

4 Legic Advant

Legic Advant ist der 2003 vorgestellte [48] direkte Nachfolger von Legic Prime, der die Probleme seines Vorgängers beheben soll. Statt auf dem proprietären Legic-RF-Protokoll (Abschnitt 3.1.3) basieren diese Karten auf dem ISO-15693- bzw. ISO-14443-Standard. Zusätzlich kommen standardisierte und öffentlich bekannte Verschlüsselungsverfahren wie DES oder AES zum Einsatz, um die Sicherheit des Systems zu gewährleisten [49]. Zudem ist, zumindest bei den ISO-14443-basierten Varianten, die Kommunikationsdistanz mit bis zu zehn Zentimetern wesentlich geringer als noch beim Vorgänger [50, S.2], was unbemerkte Angriffe auf die Funkschnittstelle erschwert.

Bislang existieren keine öffentlichen Untersuchungen zur genauen Funktionsweise und Sicherheit von Legic Advant. Daher soll im Folgenden ein Zugangskontrollsystem auf Basis dieser Technologie genauer untersucht werden.

4.1 Funkschnittstelle

Nachdem die Funkschnittstelle des Vorgängers so schlecht abgesichert war, dass das System alleine darüber vollständig kompromittiert werden konnte, soll diese beim Nachfolger als erstes näher betrachtet werden.

Dazu kommt ein handelsüblicher RFID-Leser von Touchatag [51] zum Einsatz (Abbildung 4.1), der nach dem ISO-14443-Standard mit der Karte kommuniziert. Der Leser wird per USB an den PC angeschlossen und dort mittels libnfc [52] angesprochen.

4.1.1 Identifikation des Chips

Wie sich schnell zeigt, sind die in den vorliegenden Ausweisen verwendeten Chips des Typs ATC4096 keine reine Neuentwicklung. Drei auf der libnfc basierende Tools erleichtern das Auslesen der zur Identifikation benötigten Daten.

lsnfc [53] versucht, anhand der von jeder Karte nach dem ISO-14443-Standard während der Antikollions-Phase bereitgestellten Parameter (ATQA, SAK, ATS) [54], die möglichen Chips zu identifizieren (Listing 4.1).

```
device = ACS ACR 38U-CCID 00 00 / ACR122U102 - PN532 v1.4 (0x07)
Several possible matches:
* NXP MIFARE DESFire EV1 4k
* NXP MIFARE Plus 1k
* NXP MIFARE Plus 4k
* NXP JCOP31 or JCOP41
```

Listing 4.1: Ausgabe von lsnfc: automatische Identifizierung der erkannten Tags

Abbildung 4.1: Touchatag RFID-Leser

Für die drei letzten Vorschläge sind kaum Identifikationsmerkmale hinterlegt (einzig der SAK-Wert), sodass es sich dabei wohl um falsch-positive Ergebnisse handelt. Für den ersten Vorschlag werden jedoch sämtliche Parameter überprüft und auch der manuelle Vergleich der von nfc-list [55] angezeigten Daten (Listing 4.2) mit den von NXP veröffentlichten Werten zur Identifikation der MIFARE-Tags [56, S.9ff] zeigt, dass die Einstufung als DESFire-Karte mit hoher Wahrscheinlichkeit korrekt ist.

```
nfc-list use libnfc 1.5.0 (r1019)
Connected to NFC device: ACS ACR 38U-CCID 00 00 / ACR122U102 - PN532
v1.4 (0x07)
    ATQA (SENS_RES): 03 44
      SAK (SEL_RES): 20
               ATS: 75 77  81  02  80
```

Listing 4.2: Ausgabe von nfc-list: Antikollisions-Parameter

mifare-desfire-info [57] schließlich liefert noch einige Details mehr, die auch die Unterscheidung zwischen DESFire und DESFire EV1 ermöglichen (Listing 4.3). Denn während ATQA, SAK und ATS bei beiden identisch sind, heben sich EV1-Karten durch eine Hardware- und Software-Major-Version größer Null von ihren Vorgängern ab [58][59, S.7].

```
Hardware Information:
    Vendor ID:          0x04
    Type:               0x01
    Subtype:            0x01
    Version:            1.0
    Storage size:       0x18 (=4096 bytes)
    Protocol:           0x05
Software Information:
    Vendor ID:          0x04
    Type:               0x01
    Subtype:            0x01
    Version:            1.3
    Storage size:       0x18 (=4096 bytes)
    Protocol:           0x05
```

Listing 4.3: Ausgabe von mifare-desfire-info: DESFire-spezifische Informationen

4.1.2 Angriffe

Derzeit sind für DESFire-EV1-basierte Karten keine erfolgsversprechenden Angriffe bekannt, im Gegensatz zum Vorgänger DESFire auch keine Seitenkanal-Attacken [60, S.10].

NXP bewirbt DESFire EV1 zudem mit einer Zertifizierung des BSI (Bundesamt für Sicherheit in der Informationstechnik) gemäß Common Criteria (CC) EAL4+ [61][62, S.3], welche Legic auch zumindest für die größte der Chip-Versionen, ATC4096, verspricht [49, S.2]. Das Evaluation Assurance Level (EAL) ist dabei kein Maß für die Sicherheit eines Systems, sondern für den Aufwand und die Tiefe der Zertifizierungsmaßnahmen zur Bestätigung der vorgegebenen Sicherheitsziele, in diesem Fall unter anderem spezifiziert im Smartcard IC Platform Protection Profile [63][64, S.4f].

Insbesondere diese Ziele und die getroffen Annahmen beeinflussen die Aussagekraft maßgeblich. So erhielt auch ein bekanntermaßen von zahlreichen Sicherheitslücken geplagtes Microsoft Windows XP eine EAL4+-Zertifizierung [65], unter anderem aber nur unter der im Internet-Zeitalter unrealistischen Annahme, dass keine (Netzwerk-)Verbindungen zu fremden Systemen bestehen [66, S.27]. Eine CC-Zertifizierung kann somit immer nur ein Indikator für das Einhalten gewisser Mindeststandards sein, aber keinesfalls für das Ausbleiben sämtlicher Sicherheitsprobleme.

4.1.3 Schwachstellen

Da bislang keine Angriffsmöglichkeiten bekannt sind, bleibt nur die Option, Konfigurationsfehler (beispielsweise schwache (Standard-)Schlüssel) auszunutzen, um weitere Informationen zu gewinnen. Hier helfen die oben erwähnten Tools nicht mehr weiter, mit libfreefare [67] steht jedoch eine vollständige Implementierung des DESFire-EV1-Protokolls zur Verfügung, mit der eigene Tests (Abschnitt 5.1) durchgeführt werden können.

DESFire-Karten sind in der Lage, verschiedene Schlüssel für unterschiedliche Zwecke einzusetzen (Abbildung 4.2). Auf jeder Karte existiert eine so genannte Master-Application (AID 0), über die grundlegende Einstellungen vorgenommen und weitere Anwendungen, in denen die eigentlichen Nutzdaten gespeichert sind, verwaltet werden können. Zum Zugriff auf die Anwendungen können getrennte Schlüssel vergeben werden, die sich zudem an bestimmte Berechtigungen binden lassen, zum Beispiel ein Schlüssel zum Lesen von Daten, ein anderer zum Schreiben [68, S.16].

Wie eigene Tests ergeben haben, scheint die Master-Application der vorliegenden Karten gut abgesichert. Bekannte Standard-Schlüssel (Listing 4.4), etwa ein Schlüssel nur aus Null-Bytes bestehend oder die in der AES-Spezifikation [69] veröffentlichten Schlüssel, können nicht zur Authentifizierung genutzt werden. Ohne Authentifizierung sind keine der möglichen Funktionen freigegeben, nicht einmal das Auflisten der weiteren vorhandenen Anwendungen. Wie sich zeigt, bietet letzteres jedoch keinen wirklichen Schutz.

Zwar lässt sich ohne weiteres keine Liste der vorhandenen Anwendungen abfragen (Befehl „Get Application IDs“ [70, S.8]), aber die Menge der möglichen AIDs, über die

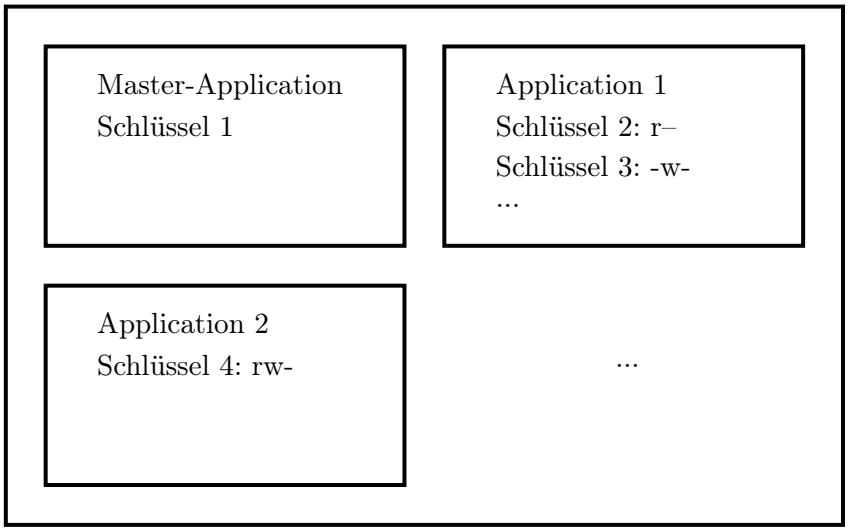

Abbildung 4.2: Beispielhafte Darstellung von verschiedenen Anwendungen und den zugehörigen Schlüsseln einer Legic-Advant-Karte

die Anwendungen identifiziert werden, ist begrenzt. Eine AID besteht aus drei Byte, entsprechend ergeben sich maximal 2^{24} verschiedene AIDs. Wählt man nun eine Anwendung, das heißt eine AID, aus (Befehl „Select Application“ [70, S.8]), um sich für diese Anwendung zu authentifizieren, antwortet die Karte mit einer Fehlermeldung, sollte die gewählte AID nicht existieren, noch bevor überhaupt ein Schlüssel zur Authentifizierung übermittelt wurde. Somit lässt sich durch einfaches Durchprobieren aller möglichen AIDs (Listing 5.6) eine Liste der für die jeweilige Karte gültigen Werte ermitteln (Listing 4.5).

```
00000000000000000000000000000000
000102030405060708090a0b0c0d0e0f
2b7e151628aed2a6abf7158809cf4f3c
```

Listing 4.4: Liste der getesteten AES-Schlüssel

Für jede AID ist nun noch zu ermitteln, welche bzw. wie viele Schlüssel hinterlegt sind, um diese anschließend durchprobieren zu können. Dies geschieht wieder durch Ausprobieren aller vierzehn möglichen Schlüsselnummern [71, S.2] für den Befehl „Get Key Version“ [70, S.8], der nicht nur die Schlüssel-Version ermittelt, sondern auch mitteilt, ob an der angefragten Position überhaupt ein Schlüssel hinterlegt ist. Allerdings sind auch die so gefundenen AIDs nicht angreifbar. Eine Authentifizierung war mit keinem der getesteten Schlüssel (Listing 4.4) möglich.

```
valid application: 0
valid application: 855
```

Listing 4.5: Liste der ermittelten Application IDs

Der Hersteller der Karten wollte leider nicht mitteilen, ob für alle Karten die gleichen Schlüssel verwendet werden, oder mittels Key Diversification für jede Karte separate

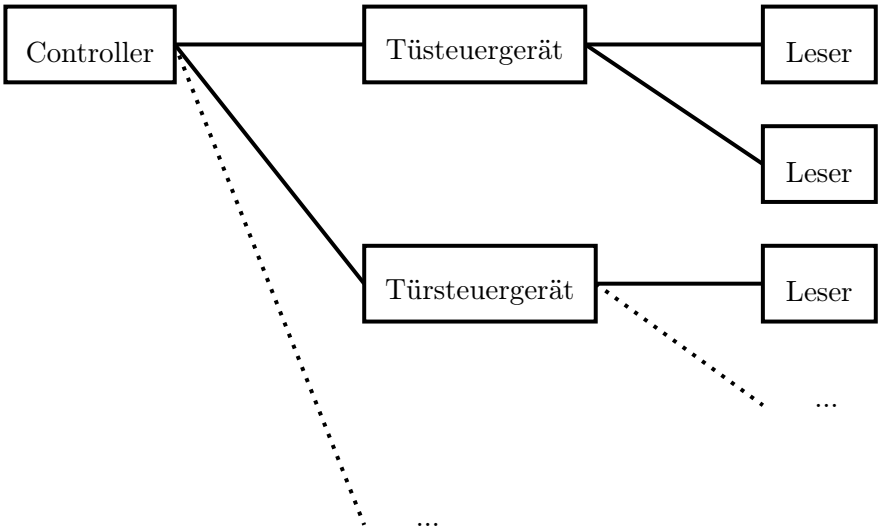

Abbildung 4.3: Architektur des Zugangskontrollsystems

Schlüssel erzeugt werden. Ohne Key Diversification genügt es einem Angreifer den Schlüssel einer einzigen Karte zu ermitteln, um Zugriff auf sämtliche eingesetzten Karten zu erhalten oder gänzlich neue zu erzeugen. Je nach Art des Angriffs liegen die Kosten einer Attacke mit tausend bis fünfzigtausend Euro [30, S.8] durchaus im realistischen Bereich.

Ebenso kritisch ist die Aussage des Herstellers zu sehen, dass mit den eingesetzten Karten keine kundenspezifischen Schlüssel verwendbar sind. Nicht nur hat der Kunde somit weder Kenntnis noch Kontrolle über die Schlüssel, es steht zudem zu befürchten, dass die gleichen Schlüssel auch bei anderen Kunden zum Einsatz kommen, ein Sicherheitsproblem dort sich also auch auf alle anderen Kunden ausweiten kann.

4.2 Stationäre Zugangskontrolle

Die stationäre Zugangskontrolle bildet das Gegenstück zu den bereits untersuchten Ausweisen und dient der Sicherung der Ein- und Ausgänge sowie weiterer Türen innerhalb der Gebäude.

4.2.1 Architektur

Es sind im wesentlichen drei getrennte Komponenten zu unterscheiden. Das Lesegerät für die Karten, ein Türsteuergerät und ein nachgeschalteter Controller (Abbildung 4.3).

Bis zu zwei Leser sind über einen gemeinsamen RS485-Bus mit einem Türsteuergerät verbunden und können jeweils eine eigene Tür kontrollieren. Jedes Türsteuergerät wiederum kommuniziert über einen separaten RS485-Bus mit dem Controller, der die Zutrittsberechtigungen der einzelnen Karten verwaltet. Erkennt ein Leser eine gültige

Karte, meldet er dies an das Steuergerät, das, ggf. nach Freigabe durch den Controller, die zugehörige Tür durch Schalten eines Relays öffnet.

4.2.2 Angriffe

Türsteuergerät und Controller sind üblicherweise in einem gesicherten Bereich untergebracht, können von einem Angreifer also nicht ohne weiteres erreicht werden. Die Angriffsmöglichkeiten beschränken sich daher auf die Manipulation des Lesers und seiner Verbindung zum Türsteuergerät.

Replay

Wie die nähere Untersuchung gezeigt hat, ist das Protokoll auf dieser seriellen Verbindung kryptografisch in keiner Weise gesichert. Dadurch genügt es beispielsweise schon, eine einzige Türöffnungsnachricht eines Lesers (Listing 4.6) aufzuzeichnen und erneut zu senden, um die Tür zu öffnen, ohne in Besitz einer gültigen Karte zu sein.

```
[63, 230, 230, 178, 122, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 54, 230, 0]
[63, 230, 230, 178, 118, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 54, 242, 0]
[63, 230, 230, 178, 114, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 54, 238, 0]
[63, 230, 230, 178, 110, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 54, 250, 0]
[63, 230, 230, 178, 106, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 54, 246, 0]
[63, 230, 230, 178, 102, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 54, 30, 0]
[63, 230, 230, 178, 98, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 54, 26, 0]
[63, 230, 230, 178, 94, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 54, 38, 0]
[63, 230, 230, 178, 90, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 54, 34, 0]
[63, 230, 230, 178, 86, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 54, 46, 0]
[63, 230, 230, 178, 82, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 54, 42, 0]
[63, 230, 230, 178, 78, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 54, 54, 0]
[63, 230, 230, 178, 74, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 54, 50, 0]
[63, 230, 230, 178, 70, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 54, 62, 0]
[63, 230, 230, 178, 66, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 54, 58, 0]
[63, 230, 230, 178, 62, 62, 58, 178, 62, 50, 58, 58, 62, 58, 242, 50, 234, 0]
```

Listing 4.6: Türöffnungsnachrichten eines Lesers im Offline-Betrieb, die sich alle 16 Versuche unabhängig von der eingesetzen Karte wiederholen

Zwar scheiterte die praktische Verifikation dieses Angriffs mit der vorhandenen Hardware bislang vermutlich an den zur Verfügung stehenden USB-RS485-Adaptern, allerdings wurde die Existenz dieser Schwachstelle selbst vom Hersteller bestätigt. Erst neuere Geräte, die dann auch die Kompatibilität zu Legic Prime aufgeben, sollen eine verschlüsselte Kommunikation unterstützen, um derartige Probleme zu vermeiden. Eine Analyse dieses Protokolls konnte mangels entsprechender Hardware nicht durchgeführt werden.

Denial of Service

Aufgrund des geteilten RS485-Busses ist ein Denial-of-Service-Angriff auf den zweiten Leser und damit die zweite Tür an einem Türsteuergerät denkbar. Dazu werden über den Anschluss des ersten Lesers ständig beliebige Daten über den Bus gesendet, sodass für alle anderen Teilnehmer keine Kommunikation mehr möglich ist.

Dieser Angriff konnte experimentell bestätigt werden, hat aber nur geringe praktische Relevanz, da nur eine weitere Tür blockiert werden kann, und das auch nur solange der Angriff aktiv durchgeführt wird. Durch die getrennten Bussysteme für die Verbindung zum Controller sind bei korrekter Verkabelung Auswirkungen auf die Leser an anderen Türsteuergeräten ausgeschlossen.

4.2.3 Schwachstellen

Die Lesegeräte sind auch im Offline-Betrieb selbst ohne angeschlossenes Türsteuergerät in der Lage, gültige von ungültigen Karten zu unterscheiden, wobei die Gültigkeit alleine daran festgemacht wird, ob der korrekte Firmencode auf der Karte hinterlegt ist. Ohne Kenntnis der verwendeten Schlüssel konnten keine Unterschiede zwischen gültigen und ungültigen Karten hinsichtlich der abfragbaren Daten ermittelt werden. Daher ist davon auszugehen, dass bereits im Leser die Schlüssel für den Zugriff auf die Karten hinterlegt sind, damit dieser diese Unterscheidung treffen kann. Möglicherweise dient dazu der Legic-Chip links unten in [i4].

Dies sollte normalerweise vermieden werden, da die Leser im ungeschützten Bereich montiert werden und daher leicht einem Angreifer in die Hände fallen können. Mitunter kann dieser dann mit moderatem finanziellen Aufwand die hinterlegten Schlüssel ähnlich wie bei der Analyse einer Karte in Erfahrung bringen. Der Unterschied besteht jedoch darin, dass hier auch Key Diversification keinen wirksamem Schutz mehr bietet, da der Leser zwangsläufig in der Lage sein muss, mit allen Karten zu kommunizieren, also über den Hauptschlüssel verfügen muss, der dann auch dem Angreifer in die Hände fiele.

Besser wäre es daher, diente der Leser lediglich als Schnittstelle ohne eigene Intelligenz, nur zur Durchleitung der Kommunikation mit den nachgeschalteten Systemen [30, S.10].

5 Analyse-Werkzeuge

Die Werkzeuge, mit denen die Erkenntnisse des vorhergehenden Kapitels ermittelt wurden, sollen in diesem Kapitel detaillierter dargestellt werden, um eine einfache Reproduktion der Ergebnisse zu ermöglichen und Möglichkeiten zur Analyse anderer Systeme aufzuzeigen.

5.1 Funkschnittstelle

Zur näheren Untersuchung der Funkschnittstelle eignen sich verschiedene Tools auf Basis der libnfc.

5.1.1 libnfc

libnfc [52] wurde Anfang 2009 [72] von Roel Verdult [73] unter der LGPL3 [74] mit dem Ziel veröffentlicht, eine freie Bibliothek für RFID- bzw. NFC-Anwendungen zu schaffen. Mittlerweile unterstützt sie diverse Lesegeräte [75] auf Basis des PN35x-Chipsatzes von NXP [76] sowie verschiedene Tags, unter anderem nach dem ISO-14443-Standard. Dabei ist es über kompatible Lesegeräte sowohl möglich, Karten auszulesen, als auch Karten gegenüber einem anderen Lesegerät zu emulieren [77]

libnfc bringt einige Beispielanwendungen mit, die die Verwendung der Bibliothek demonstrieren sollen. Ein nützliches Werkzeug darunter ist nfc-list [55], das alle ISO-14443-A-kompatiblen Tags im Einzugsbereich des Lesers zusammen mit deren während der Antikollisions-Phase übermittelten Werten (ATQA, SAK, ATS) anzeigt (Listing 4.2). Anhand dieser Werte kann der Typ des Tags oftmals genau ermittelt oder zumindest stark eingegrenzt werden [54].

5.1.2 nfc-tools

Das Projekt nfc-tools [78] stellt eine Reihe von Software-Komponenten basierend auf libnfc bereit. Neben nfcutils und libfreefare (siehe unten) gibt es beispielsweise mit pam_nfc [79] die Möglichkeit, ein Tag zur Authentifizierung am Computer zu verwenden, oder mit dem Mifare Classic Offline Cracker, kurz MFOC [80], die Schlüssel von Mifare-Classic-Karten zu ermitteln.

nfcutils

nfcutils [53] besteht nur aus einer einzigen Anwendung, lsnfc, die ähnlich wie nfc-list die Antikollisions-Werte der Tags ermittelt (Listing 4.1). Diese werden mit einer Liste bekannter Tag-Typen abgeglichen, um sie einem menschenlesbaren Produktnamen zuordnen zu können. Für einzelne Typen, etwa Mifare DESFire, sind weitere Algorithmen

hinterlegt, die anhand zusätzlicher, nicht zwingend standardisierter Informationen, den genauen Sub-Typ ermitteln und so beispielsweise zwischen DESFire und DESFire EV1 unterscheiden.

libfreefare

libfreefare [81] implementiert das von verschiedenen Mifare-Karten (u.a. Classic, DESFire, Ultralight) verwendete Protokoll und stellt eine entsprechende high-level API zur Interaktion mit diesen Karten bereit. Ein Werkzeug, das speziell für DESFire-basierte Karten mitgeliefert wird, ist mifare-desfire-info [57]. Dadurch, dass es das DESFire-spezifische Protokoll verwendet, kann es mehr Informationen über die erkannten Tags bereitstellen (Listing 4.3), als die bislang beschrieben nfc-list beziehungsweise lsnfc.

Zu diesen Informationen zählt beispielsweise das Produktionsdatum oder die Software- und Hardware-Version, anhand derer DESFire- von DESFire-EV1-Karten unterschieden werden können [58][59, S.7]. Des Weiteren wird versucht, diverse Sicherheitseinstellungen der Karte auszulesen, etwa ob auch ohne Kenntnis des Hauptschlüssels neue Anwendungen erstellt oder vorhandene aufgelistet werden können. Abschließend erfolgt, sofern möglich, eine Anzeige des noch auf der Karte zur Verfügung stehenden freien Speichers.

5.1.3 Eigenentwicklung

Für verschiedene Tests (Abschnitt 4.1.3) wurden auf Basis von libfreefare eigene Python-Anwendungen erstellt. Obwohl libfreefare selbst in C geschrieben ist, fiel die Wahl auf Python, eignen sich C-Anwendungen doch aufgrund des aufwändigeren Codes (zum Beispiel bedingt durch manuelle Speicherverwaltung) und das nach jeder Änderung notwendige Kompilieren nur bedingt für flexibles Ausprobieren unterschiedlicher Ansätze. Dies macht es jedoch zunächst notwendig, Bindings zu erzeugen, um von Python auf die C-Bibliothek zugreifen zu können.

Python-Bindings

Zur Erstellung der Bindings gibt es mit Python unterschiedliche Möglichkeiten, unter anderem das Python-eigene Modul ctypes [82], mit dem Bibliotheksfunktionen über reinen Python-Code eingebunden und aufgerufen werden können. Da ctypes jedoch kaum Kenntnisse über die aufgerufenen Funktionen und die verwendeten Datentypen hat, muss Code dafür gegebenenfalls manuell erzeugt werden, beispielsweise für den Zugriff auf structs. Dies macht den Prozess, insbesondere wenn viele verschiedene Funktionen und komplexe Datentypen verwendet werden sollen, aufwändig und fehlerträchtig.

Komfortabler ist daher die Verwendung eines Tools wie SWIG (Simplified Wrapper and Interface Generator) [83], das einen Großteil des Prozesses automatisiert erledigen kann. SWIG unterstützt C- und C++-Bibliotheken und kann diese an diverse Skriptsprachen, so auch Python, anbinden. SWIG greift auf die mit den Bibliotheken ausgelieferten Header-Dateien zurück, und hat daher schon mehr Informationen zur Verfügung als ctypes. Zusätzlich wird vom Entwickler eine kurze Schnittstellen-Beschreibung für SWIG erzeugt, in der die zu verwendenden Funktionen aufgelistet werden und Typkonvertierungen, etwa

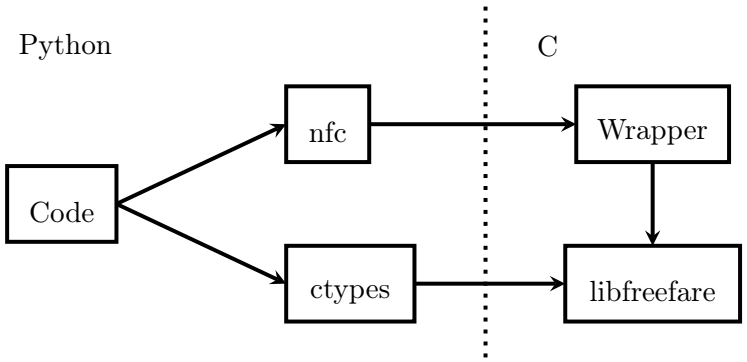

Abbildung 5.1: Möglichkeiten für Python-Bindings: SWIG (oben) und ctypes (unten)

für Arrays, sowie weitere Anpassungen festgelegt werden können. Im Gegensatz zu ctypes generiert SWIG eine zusätzliche Schicht C-Code, die die Anbindung an die Bibliothek vornimmt, sowie ein passendes Modul für die Zielsprache, in diesem Fall Python, das mit dem C-Wrapper im Hintergrund den Zugriff für Python-Anwendungen bereitstellt (Abbildung 5.1).

Die Schnittstellen-Beschreibung, mit der SWIG alles weitere selbstständig erzeugt, setzt sich wie folgt zusammen. Zunächst wird der Name des zu erzeugenden Moduls deklariert, dann einige Typumwandlungen definiert und schließlich die benötigten Header-Dateien der Bibliothek in den generierten C-Code eingebunden (Listing 5.1). Zwei Typen werden dabei mit *OUTPUT* als Ausgabe-Parameter deklariert, sodass SWIG dafür sorgt, dass deren Werte als Rückgabewerte der Python-Funktion erscheinen, auch wenn sie in der Funktion der C-Bibliothek eigentlich Übergabe-Parameter sind.

```
%module nfc

%include "typemaps.i"
%apply unsigned int { uint8_t };
%apply unsigned int { uint32_t };
%apply unsigned int *OUTPUT { uint8_t *version };
%apply unsigned int *OUTPUT { size_t *pszDeviceFound };

%{
#include <nfc/nfc-types.h>
#include <nfc/nfc.h>
#include <freefare.h>
%}
```

Listing 5.1: SWIG-Schnittstellen-Beschreibung für libfreefare (Teil 1)

Mit dem folgenden Konstrukt (Listing 5.2) können Fehler, die durch die Bibliotheksfunktionen anhand ihres Rückgabewerts signalisiert werden, in Python-Exceptions umgewandelt werden. Anschließend folgt eine Liste der gewünschten Funktionen, auf die diese Überprüfung angewendet wird.

```
%define CHECK_RESULT(func)
%exception func {
```

```
  $action
  if (result < 0) {
     PyErr_SetString(PyExc_RuntimeError, freefare_strerror(arg1));
     return NULL;
  }
}
%enddef

CHECK_RESULT(mifare_desfire_authenticate)
CHECK_RESULT(mifare_desfire_connect)
CHECK_RESULT(mifare_desfire_disconnect)
CHECK_RESULT(mifare_desfire_get_key_version)
CHECK_RESULT(mifare_desfire_select_application)
```

Listing 5.2: SWIG-Schnittstellen-Beschreibung für libfreefare (Teil 2)

Schließlich werden die Header-Dateien auch in die Schnittstellen-Beschreibung selbst eingebunden (Listing 5.3), um anschließend einige Typen zusätzlich in Form von Arrays beziehungsweise Pointern zur Verfügung zu stellen. SWIG wird für diese speziellen Python-Code zum leichten Zugriff generieren, etwa das im Grundgerüst unten verwendete *nfc.mifare_tag_array.frompointer*.

```
%include <nfc/nfc-types.h>
%include <nfc/nfc.h>
%include <freefare.h>

%include "carrays.i"
%array_class(nfc_device_desc_t, device_desc_array);
%array_class(MifareTag, mifare_tag_array);
%array_class(MifareDESFireDF, mifare_df_array);
%array_class(uint8_t, uint8_array);

%include "cpointer.i"
%pointer_functions(MifareDESFireDF*, dfpointer);
%pointer_functions(uint8_t*, uint8_pointer);
%pointer_functions(size_t, size_pointer);
%pointer_functions(size_t*, size_ppointer);
```

Listing 5.3: SWIG-Schnittstellen-Beschreibung für libfreefare (Teil 3)

Grundgerüst

Basierend auf den von SWIG generierten Python-Bindings lässt sich ein kurzes Programm (Listing 5.4) entwickeln, das alle Karten auf allen angeschlossenen Lesegeräten ermittelt und eine Verbindung herstellt, um nach Bedarf weitere Befehle an die Karte abzusetzen.

```
import nfc
devices = nfc.device_desc_array(8)
device_count = nfc.nfc_list_devices(devices, 8)
for i in range(device_count):
        device = nfc.nfc_connect(devices[i])
        tags = nfc.mifare_tag_array.frompointer(nfc.freefare_get_tags(device))
        j = 0
        while tags[j]:
                print(nfc.freefare_get_tag_uid(tags[j]))
                nfc.mifare_desfire_connect(tags[j])
                # handle tag
                nfc.mifare_desfire_disconnect(tags[j])
```

```
            j += 1
        nfc.freefare_free_tags(tags.cast())
        nfc.nfc_disconnect(device)
```

Listing 5.4: Python-Grundgerüst zum Zugriff auf eine Mifare-DESFire-Karte

Die folgenden Beispiele lassen sich alle in obiges Grundgerüst einbetten.

Zugriff auf die Master-Application

Für den Zugriff auf die Master-Application wird üblicherweise ein Schlüssel benötigt, der in einem Byte-Array abgelegt wird. Zusätzlich ist es über die Schlüssel-Version möglich, einen von mehreren hinterlegten Schlüsseln zur Authentifizierung auszuwählen. Folgender Code (Listing 5.5) versucht die Authentifizierung an der Master-Application mit einem nur aus Null-Bytes bestehenden Schlüssel.

```
key_data_aes = nfc.uint8_array(16)
for i in range(16):
        key_data_aes[i] = 0
key_no = 0
version = nfc.mifare_desfire_get_key_version(tags[j], key_no)[1]
key = nfc.mifare_desfire_aes_key_new_with_version(key_data_aes, version)
nfc.mifare_desfire_authenticate(tags[j], key_no, key)
```

Listing 5.5: Authentifizierung an der Master-Application mit Schlüssel aus Null-Bytes

Auflistung der vorhandenen Application-IDs

Zur Auflistung der auf der Karte vorhandenen Application-IDs steht normalerweise der Befehl *mifare_desfire_get_application_ids* zur Verfügung. Bei entsprechender Konfiguration der Karte kann dieser jedoch nur nach vorhergehender Authentifizierung mit dem Hauptschlüssel verwendet werden.

Der Befehl *mifare_desfire_select_application*, mit dem eine einzelne Anwendung für den weiteren Zugriff ausgewählt werden kann, erzeugt jedoch einen Fehler, sollte die angeforderte Application-ID nicht existieren, und kann daher verwendet werden, um alle möglichen Application-IDs durchzuprobieren (Listing 5.6) und so die gültigen Werte zu ermitteln (Listing 4.5).

```
for aid in range(0, 0xffffff):
        try:
                nfc.mifare_desfire_select_application(tags[j], nfc.mifare_desfire_aid_new(aid))
        except:
                pass
        else:
                print('valid application: {0}'.format(aid))
```

Listing 5.6: Auflisten der vorhandenen Application-IDs (Brute-Force)

Zugriff auf weitere Anwendungen

Ist die ID einer weiteren Anwendung auf der Karte bekannt, kann ähnlich wie bei der Master-Application die Authentifizierung mit einem anwendungsspezifischen Schlüssel

vorgenommen werden. Der folgende Code (Listing 5.7) gleicht daher diesem Fall, wählt aber zunächst noch eine Anwendung mit der ID 123 aus.

```
nfc.mifare_desfire_select_application(tags[j], nfc.mifare_desfire_aid_new(123))
key_data_aes = nfc.uint8_array(16)
for i in range(16):
       key_data_aes[i] = 0
version = nfc.mifare_desfire_get_key_version(tags[j], 0)[1]
key = nfc.mifare_desfire_aes_key_new_with_version(key_data_aes, version)
nfc.mifare_desfire_authenticate(tags[j], version, key)
```

Listing 5.7: Authentifizierung an Anwendung 123 mit Schlüssel aus Null-Bytes

5.2 Serielle Schnittstelle

Da auf der seriellen Schnittstelle der Leser kein bekanntes Protokoll verwendet wird, bleibt hier nur die Möglichkeit, mit einer Eigenentwicklung die auf der Leitung ausgetauschten Byteströme mitzulesen und auszuwerten. Auch für diese Aufgabe bietet sich eine Python-Anwendung auf Basis von pyserial [85] an.

5.2.1 Mitlesen von Daten

Zum Mitlesen der ausgetauschten Daten (Listing 5.8) genügt ein USB-RS-485-Adapter, der zusätzlich zu Leser und Steuergerät an die beiden Leitungen des RS-485-Busses angeschlossen wird (Abbildung 5.2).

```
import serial
device = serial.Serial('/dev/ttyUSB0')
while True:
       print(device.read(1).encode('hex'))
```

Listing 5.8: Ausgabe des Datenstroms einer seriellen Schnittstelle

Mitunter lassen sich in diesem Datenstrom einzelne logische Blöcke erkennen, etwa durch die zeitliche Abfolge der Datenübertragung oder durch Terminatoren, zum Beispiel die häufig für diesen Zweck verwendeten Null-Bytes. Folgender Code (Listing 5.9) nutzt beide Möglichkeiten um vermutete Datenpakete ([l20], Listing 4.6) zu erkennen. Alle Bytes, die maximal zwei Millisekunden nach dem vorherigen eingehen, werden als zusammengehörig betrachtet. Ein Null-Byte dient zusätzlich als Trennzeichen.

```
def receive(dev):
       result = ''
       while True:
              try:
                     c = dev.read(1)
              except TypeError:
                     return result
              else:
                     result += c
                     if c == '\x00':
                            return result

import serial
```

```
device = serial.Serial('/dev/ttyUSB0', timeout=0.002)
while True:
        result = receive(device)
        if result:
                print([ord(x) for x in result])
```

Listing 5.9: Aufteilung des Datenstroms einer seriellen Schnittstelle

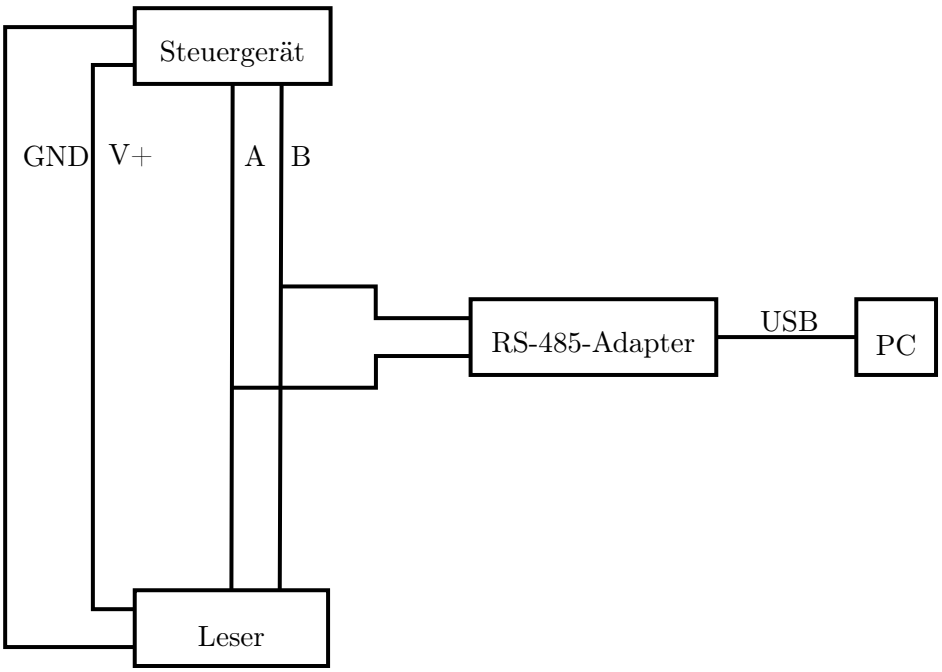

Abbildung 5.2: Mitlesen der auf der RS-485-Schnittstelle ausgetauschten Daten

5.2.2 Imitieren von Geräten

Statt nur Daten mitzulesen, können auch Daten gesendet werden, sobald bestimmte Pakete empfangen wurden (Listing 5.10), um so eins der teilnehmenden Geräte zu imitieren (Abbildung 5.3). Wie sich gezeigt hat, muss dies jedoch nicht immer erfolgreich sein, beispielsweise da das empfangende Gerät sehr strikte Anforderungen hat, in welchem zeitlichen Abstand eine Antwort eingehen muss.

```
import serial
device = serial.Serial('/dev/ttyUSB0', timeout=0.002)
while True:
        result = receive(device)
        if result == '\x10\x20\x30\x00':
                device.write('\x30\x20\x10\x00')
```

Listing 5.10: Imitieren einer Antwort auf eingehende Datenpakete

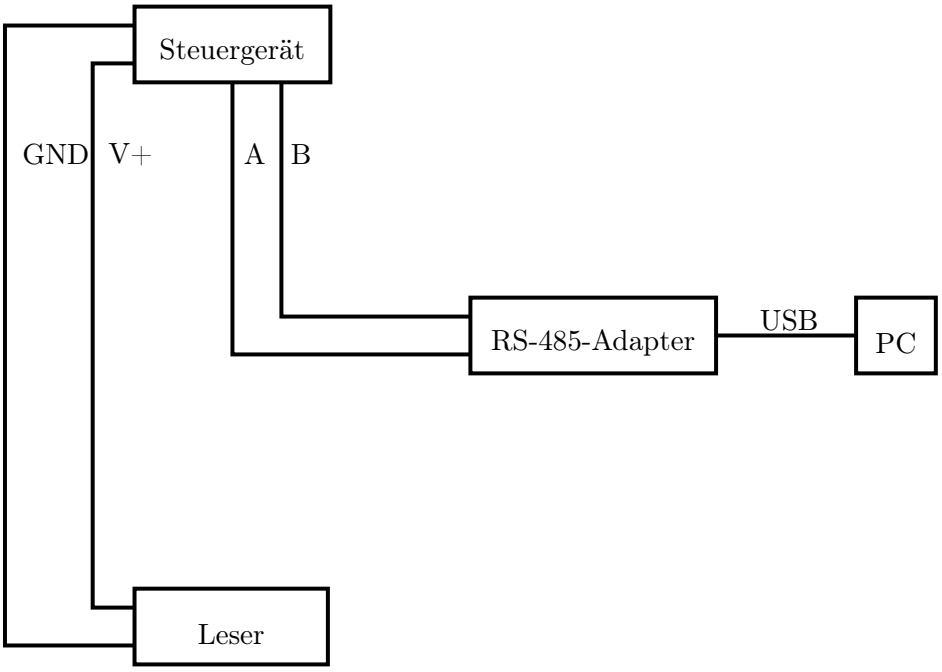

Abbildung 5.3: Imitieren des Lesers gegenüber dem Steuergerät

5.2.3 Man-in-the-middle-Angriff

Wäre das Senden von Daten problemlos möglich, ließe sich durch den Einsatz von zwei USB-RS-485-Adaptern (Abbildung 5.4) auch ein Man-in-the-middle-Angriff realisieren, bei dem sämtliche ausgetauschten Daten nicht nur mitgelesen, sondern bei Bedarf auch verändert werden können (Listing 5.11).

```
import serial
device1 = serial.Serial('/dev/ttyUSB0', timeout=0.002)
device2 = serial.Serial('/dev/ttyUSB1', timeout=0.002)
while True:
        result1 = receive(device1)
        result2 = receive(device2)
        if result1:
                device2.write(result1)
        if result2:
                device1.write(result2)
```

Listing 5.11: Man-in-the-middle-Angriff ohne Veränderung der ausgetauschten Daten

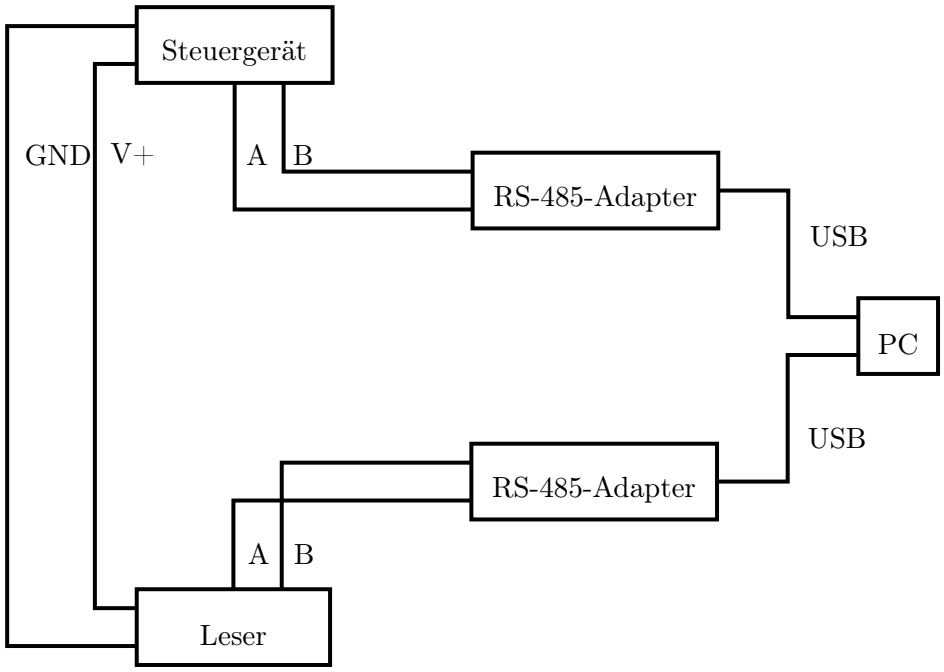

Abbildung 5.4: Man-in-the-middle-Angriff mittels zweier USB-RS-485-Adapter

6 Fazit

Wie gezeigt wurde, nimmt die Verbreitung von Smartcard-Technologien in vielen Bereichen stetig zu, können sie doch bei korrekter Implementierung einen deutlichen Komfort für den Anwender auf gleichzeitig hohem Sicherheitsniveau bieten. Dennoch musste festgestellt werden, dass viele, auch weit verbreitete, Systeme, ihre Versprechen nicht halten und eklatante Sicherheitslücken aufweisen. Die Gründe dafür sind vielfältig, sei es, weil Produkte nicht mehr dem Stand der Technik entsprechen oder aus Kostengründen auf die günstigsten Modelle zurückgegriffen wird. Erst langsam setzt sich die Erkenntnis durch, dass Security by Obscurity, das Geheimhalten aller Informationen über ein System, oft kein ausreichender Schutz gegen entschlossene Angreifer ist, und grundlegende Algorithmen besser durch frühzeitige Veröffentlichung auf ihre Wirksamkeit überprüft werden sollten.

Ein besonderes Negativ-Beispiel stellt Legic Prime dar, dessen Verfahren Legic RF trotz entsprechender Bezeichnung keinerlei Verschlüsselung der Kommunikation vornimmt, da nirgends geheime Schlüssel verwendet werden. Somit ist es möglich, nur durch Nachvollziehen des umfangreich verschleierten Protokolls, sowohl Leser als auch Karten beliebig zu emulieren. Dies erlaubt es zudem, weitere Sicherheitsmechanismen wie den Schreib- und Leseschutz von Kartendaten zu umgehen, da derartige Beschränkungen lediglich von der Legic-Software beachtet, aber nicht von der Karte selbst durchgesetzt werden. So zeigt sich, dass Legic Prime nur deswegen lange Zeit als sicher galt, weil sich niemand intensiver mit seiner Funktionsweise beschäftigt hat. Aufgrund der zahlreichen Probleme sollte es heute nicht mehr verwendet werden.

Die Bewertung des Nachfolgers Legic Advant muss schließlich zweigeteilt ausfallen. Geht es lediglich darum, die gröbsten Sicherheitsprobleme von Legic Prime auszuräumen, ist es sicherlich ein passables Upgrade, insbesondere da die Migration dank der Abwärtskompatibilität schrittweise im Hintergrund erfolgen kann. Es werden keine proprietären, und wie sich damit in der Vergangenheit oft gezeigt hat, unsichere kryptografische Verfahren mehr eingesetzt, sondern standardisierte und geprüfte Algorithmen wie DES und AES. Auch die zugrundeliegende Hardware der Chipkarte in Form von Mifare DESFire EV1 lässt dank Common-Criteria-Zertifizierung ein gewisses Sicherheitsniveau erwarten und hat sich bislang auch fortgeschrittenen Angriffstechniken erfolgreich widersetzt. Ein Manipulieren oder Klonen von Karten als gefährlichster Angriff scheint damit derzeit nicht möglich.

Dennoch ist das untersuchte Legic-Advant-System keineswegs perfekt. Insbesondere der stationäre Teil der Zugangskontrolle offenbart große Schwächen, ist die Kommunikation zwischen den einzelnen Komponenten abgesehen von der fehlenden Dokumentation kaum geschützt. Zudem sind die kryptografischen Schlüssel, die Zugang zu allen Karten des Systems ermöglichen, direkt im frei zugänglichen Leser hinterlegt, und können einem Angreifer somit wesentlich einfacher in die Hände fallen, als wenn sie im geschützten

Bereich hinterlegt wären und der Leser nur die Kommunikation durchleitet.

Ein weiteres Problem liegt darin, dass sich das Schlüsselmaterial vollständig unter Kontrolle des Herstellers befindet und dieser nicht bereit ist, dessen Sicherungsmaßnahmen zu erläutern. So ist es ungeklärt, ob Key Diversification zum Einsatz kommt, oder ob das Extrahieren des Schlüssels aus einer Karte bereits den Zugang zu allen anderen öffnet. Auch ist nicht klar, ob ein getrennter (Haupt-)Schlüssel je Kunde verwendet wird, oder sich alle Kundenschlüssel letztlich auf das gleiche Geheimnis zurückführen lassen und so ein Sicherheitsvorfall an anderer Stelle möglicherweise auch das eigene System bedrohen kann.

Weitere Untersuchungen sind noch für den stationären Teil der Zugangskontrolle notwendig. Zwar konnte hier gezeigt werden, dass die Kommunikation zwischen Leser und Backend nicht geschützt ist, ein erfolgreicher Angriff ließ sich aber nicht demonstrieren. Zudem sollen laut Herstelleraussage neuere Leser unter Aufgabe der Abwärtskompatibilität zu Legic Prime in der Lage sein, ein sichereres Protokoll zu verwenden, dessen Qualität sich mangels weiterer Angaben erst in der Praxis beweisen muss.

Literaturverzeichnis

[1] Feld, S. / Hertlein, M. (2010): Sicherheit durch und Sicherheit bei SmartCards, http://www.internet-sicherheit.de/fileadmin/docs/publikationen/feld/Sicherheit-durch-und-Sicherheit-bei-SmartCards-iX-Forum-Feld-Hertlein.pdf

[2] Rankl, W. / Effing, W. (2002): Handbuch der Chipkarten: Aufbau – Funktionsweise – Einsatz von Smart Cards, Carl Hanser Verlag

[3] Petri, S. (1999): An Introduction to Smart Cards, http://artofconfusion.org/smartcards/docs/intro.pdf

[4] Winter, M. (2003): Telefonieren mit buntem Plastik: Die Telefonkarte wurde 20, http://www.teltarif.de/arch/2003/kw39/s11589.html

[5] 3268zauber (2009): File:Öffentliches Telefon.JPG, https://commons.wikimedia.org/wiki/File:%C3%96ffentliches_Telefon.JPG

[6] Winter, M. (2003): Neues Strukturkonzept für öffentliche Telefone, http://www.teltarif.de/arch/2003/kw18/s10479.html

[7] Nightflyer (2007): Datei:Telefonkarte - P.jpg, https://de.wikipedia.org/w/index.php?title=Datei:Telefonkarte_-_P.jpg

[8] Horvat, C. (2009): File:Ec-bankkarte.jpg, https://commons.wikimedia.org/wiki/File:Ec-bankkarte.jpg

[9] Sparkassen-Finanzgruppe (o. J.): Die GeldKarte in der Kantine - Zahlverfahren Nummer 1 für Unternehmen, https://www.scard.de/firmenkunden/geldkarte/kantine/index.htm

[10] Photocapy (2006): File:WM06 ASA-UKR Ticket.jpg, https://commons.wikimedia.org/wiki/File:WM06_ASA-UKR_Ticket.jpg

[11] Transport for London (o. J.): What is Oyster?, http://www.tfl.gov.uk/tickets/14836.aspx

[12] Queen's Printer for Ontario (2011): PRESTO Explained, https://www.prestocard.ca/en/StaticContent/PrestoExplained/

[13] Murmann, F. (2008): File:Oystercard.jpg, https://commons.wikimedia.org/wiki/File:Oystercard.jpg

[14] Aladdin (2009): File:EToken 6 models.jpg, https://commons.wikimedia.org/wiki/File:EToken_6_models.jpg

[15] Almekinders, A. (2010): File:AstonCam.jpg, https://commons.wikimedia.org/wiki/File:AstonCam.jpg

[16] Giesecke & Devrient GmbH (2011): SIM Cards, http://www.gi-de.com/en/products_and_solutions/products/mobile_communication/sim-cards.jsp

[17] Qurren (2006): File:NTT DoCoMo FOMA card chip green.jpg, https://commons.wikimedia.org/wiki/File:NTT_DoCoMo_FOMA_card_chip_green.jpg

[18] Bundesamt für Sicherheit in der Informationstechnik (o. J.): Der elektronische Reisepass, https://www.bsi.bund.de/cln_183/DE/Themen/ElektronischeAusweise/ePass/epass_node.html

[19] Bundesministerium des Innern (2011): Der neue Personalausweis - Unterschriftsfunktion, http://www.personalausweisportal.de/DE/Die_neuen_Funktionen/Unterschriftsfunktion/unterschriftsfunktion_node.html

[20] Bundesministerium des Innern (2010): File:Mustermann nPA.jpg, https://commons.wikimedia.org/wiki/File:Mustermann_nPA.jpg

[21] Bundesrepublik Deutschland (1988): § 291 Krankenversichertenkarte in Fünftes Buch Sozialgesetzbuch - Gesetzliche Krankenversicherung - (Artikel 1 des Gesetzes vom 20. Dezember 1988, BGBl. I S. 2477), zuletzt geändert durch Art. 3 G v. 22.12.2011 (BGBl. I S. 3057), http://www.gesetze-im-internet.de/sgb_5/__291.html

[22] Bundesrepublik Deutschland (1988): § 291a Elektronische Gesundheitskarte in Fünftes Buch Sozialgesetzbuch - Gesetzliche Krankenversicherung - (Artikel 1 des Gesetzes vom 20. Dezember 1988, BGBl. I S. 2477), zuletzt geändert durch Art. 3 G v. 22.12.2011 (BGBl. I S. 3057), http://www.gesetze-im-internet.de/sgb_5/__291a.html

[23] Bundesrepublik Deutschland (2001): § 2 Begriffsbestimmungen in Signaturgesetz vom 16. Mai 2001 (BGBl. I S. 876), zuletzt geändert durch Art. 4 G v. 17.7.2009 (BGBl. I S. 2091), http://www.gesetze-im-internet.de/sigg_2001/__2.html

[24] Bundesrepublik Deutschland (2002): § 126a Elektronische Form in Bürgerliches Gesetzbuch in der Fassung der Bekanntmachung vom 2. Januar 2002 (BGBl. I S. 42, 2909; 2003 I S. 738), zuletzt geändert durch Art. 1 G v. 27.7.2011 (BGBl. I S. 1600), http://www.gesetze-im-internet.de/bgb/__126a.html

[25] Bundesrepublik Deutschland (2005): § 17 Produkte für qualifizierte elektronische Signaturen in Signaturgesetz vom 16. Mai 2001 (BGBl. I S. 876), zuletzt geändert durch Art. 4 G v. 17.7.2009 (BGBl. I S. 2091), http://www.gesetze-im-internet.de/sigg_2001/__17.html

[26] Schmid, W. (2000): Arbeitszeiterfassung mit Uhren - Ein historischer Rückblick, http://www.kontrolluhren.de/download/zeiterfassung.pdf

[27] Dacs / WhiteTimberwolf (2009): File:SmartCardPinout.svg, https://commons.wikimedia.org/wiki/File:SmartCardPinout.svg

[28] Ormont, J. (2010): File:SIM chip structure and packaging.svg, https://commons.wikimedia.org/wiki/File:SIM_chip_structure_and_packaging.svg

[29] Schneier, B. (1999): Cryptography: The Importance of Not Being Different in Crypto-Gram Newsletter, http://www.schneier.com/crypto-gram-9904.html#different

[30] Rohr, A. / Nohl, K. / Plötz, H. (2010): Establishing Security Best Practices in Access Control, http://srlabs.de/blog/wp-content/uploads/2010/09/Access_Control_Best_Pratices_Study_v1.0.pdf

[31] Nohl, K. / Plötz, H. (2007): Mifare, http://mirror.fem-net.de/CCC/24C3/mp4/24c3-2378-en-mifare_security-COMPATIBLE.mp4

[32] Nohl, K. et al. (2008): Reverse-Engineering a Cryptographic RFID Tag at USENIX Security Symposium, http://www.cs.virginia.edu/evans/pubs/usenix08/usenix08.pdf

[33] Plötz, H. (2008): Mifare Classic – Eine Analyse der Implementierung, http://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2008-21/SAR-PR-2008-21_.pdf

[34] Courtois, N. / Nohl, K. / O'Neil, S. (2008): Algebraic Attacks on the Crypto-1 Stream Cipher in MiFare Classic and Oyster Cards, http://eprint.iacr.org/2008/166.pdf

[35] Courtois, N. (2009): The Dark Side of Security by Obscurity and Cloning MiFare Classic Rail and Building Passes, Anywhere, Anytime, http://eprint.iacr.org/2009/137.pdf

[36] Oswald, D. / Paar, C. (2011): Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World - Extended Version, http://www.emsec.rub.de/media/crypto/veroeffentlichungen/2011/10/10/desfire_2011_extended_1.pdf

[37] LEGIC Identsystems AG (2011): Über LEGIC Identsystems AG, http://www.legic.com/ueber_legic.html

[38] LEGIC Identsystems AG (2011): Einführung in die LEGIC prime Produktlinie, http://www.legic.com/de/legic_prime.html

[39] LEGIC Identsystems AG (2005): LEGIC News 2-2005, http://www.legic.com/user_content/editor/files/LEGIC%20News/legic_news_2_05_de.pdf

[40] Kremp, M. (2010): Alarmierende Sicherheitslücke: Hacker knacken Flughafen-Zugangskontrolle, http://www.spiegel.de/netzwelt/netzpolitik/0,1518,671980,00.html

[41] LEGIC Identsystems AG (2003): LEGIC prime Datenträger (MIM), http://www.legic.com/user_content/editor/files/LEGIC%20prime%20info/legic_prime_mim_zusfassung_de.pdf

[42] Plötz, H. / Nohl, K. (2010): Legic Prime – Obscurity in Depth at EUSecWest, http://srlabs.de/blog/wp-content/uploads/2010/07/100616.EUSecWest.LegicPrime.pdf

[43] Plötz, H. / Nohl, K. (2011): Peeling Away Layers of an RFID Security System at Financial Cryptography and Data Security '11, http://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2011-03/SAR-PR-2011-03_.pdf

[44] EVIS AG (2005): LEGIC Medien Übersicht, http://www.evis.ch/fileadmin/content/download/Legic_Merkbl_tter/LEGIC_Medien_Uebersicht.pdf

[45] LEGIC Identsystems AG (2010): Master-Token basierte Sicherheitskontrolle, http://www.legic.com/de/download_statistics.html?content.filename=mtsc-b_de-web_20100914132824.pdf

[46] Plötz, H. / Nohl, K. (2009): Legic Prime: Obscurity in Depth at 26C3, http://events.ccc.de/congress/2009/Fahrplan/attachments/1506_legic-slides.pdf

[47] Plötz, H. / Nohl, K. (2009): Legic Prime: Obscurity in Depth, http://mirror.fem-net.de/CCC/26C3/mp4/26c3-3709-en-legic_prime_obscurity_in_depth.mp4

[48] LEGIC Identsystems AG (2003): World premiere from LEGIC - the all new LEGIC advant^{TM} contactless smart card system, http://www.presseportal.ch/de/meldung/100460999/

[49] LEGIC Identsystems AG (2010): LEGIC advant Crypto Transponder Chips, http://www.legic.com/de/download_statistics.html?content.filename=la-11-006e-%28transponder-overview%29_de-web_20100914121546.pdf&content.cid=5443

[50] LEGIC Identsystems AG (2009): Kommunikationsdistanzen in kontaktlosen Smart Card Systemen, http://www.legic.com/de/download_statistics.html?content.filename=la-11-020a-de_white_paper_comm_distance_20090513134340.pdf

[51] o. V. (2009): touchatag starter pack, http://store.touchatag.com/acatalog/touchatag_starter_pack.html

[52] Verdult, R. (2009): libnfc.org - Public platform independent Near Field Communication (NFC) library, http://www.libnfc.org/documentation/introduction

[53] Conty, R. (2010): nfcutils - provide a simple 'lsnfc command that list tags which are in your NFC device field', https://code.google.com/p/nfc-tools/wiki/nfcutils

[54] o. V. (2011): libnfc.org - Tags - ISO14443, http://www.libnfc.org/documentation/hardware/tags/iso14443

[55] Conty, R. (2010): libnfc.org - Example - list, http://www.libnfc.org/documentation/examples/nfc-list

[56] NXP Semiconductors (2011): AN10833 - MIFARE Type Identification Procedure, http://www.nxp.com/documents/application_note/AN10833.pdf

[57] Tartiere, R. (2011): nfc-tools - mifare-desfire-info.c (r751), https://code.google.com/p/nfc-tools/source/browse/trunk/libfreefare/examples/mifare-desfire-info.c?r=751

[58] Conty, R. (2010): nfc-tools - lsnfc.c (r636), https://code.google.com/p/nfc-tools/source/browse/trunk/nfcutils/src/lsnfc.c?spec=svn900&r=636#130

[59] NXP Semiconductors (2011): AN11004 - MIFARE DESFire as Type 4 Tag, http://www.nxp.com/documents/application_note/AN11004.pdf

[60] Kasper, T. / Oswald, D. / Paar, C. (2011): Side-Channel Analysis of Cryptographic RFIDs with Analog Demodulation, http://rfid-cusp.org/rfidsec/files/RFIDSec2011DraftPapers/KasperEtAl.pdf

[61] NXP Semiconductors (2009): NXP's MIFARE DESFire EV1 Technology Receives Trusted Security Stamp of Approval, http://www.nxp.com/news/press-releases/2009/06/nxp-s-mifare-desfire-ev1-technology-receives-trusted-security-stamp-of-approval.html

[62] Bundesamt für Sicherheit in der Informationstechnik (2011): Certification Report BSI-DSZ-CC-0712-2011, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Reporte07/0712a_pdf.pdf?__blob=publicationFile

[63] o. V. (2001): Smartcard IC Platform Protection Profile, http://www.commoncriteriaportal.org/files/ppfiles/ssvgpp01.pdf

[64] NXP Semiconductors (2011): MIFARE DESFire EV1 MF3ICD81 - Security Target Lite, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Reporte07/0712b_pdf.pdf?__blob=publicationFile

[65] Chokhani, S. et al. (2007): Validation Report - Microsoft Windows 2003 Server and XP Workstation, http://www.commoncriteriaportal.org/files/epfiles/st_vid9506-vr.pdf

[66] Microsoft Corporation (2007): Microsoft Windows 2003/XP Security Target, http://www.commoncriteriaportal.org/files/epfiles/st_vid9506-st.pdf

[67] Tartiere, R. (2010): libfreefare - Library for high level manipulation of MIFARE cards, http://code.google.com/p/nfc-tools/wiki/libfreefare

[68] Philips Semiconductors (2004): Functional Specification - mifare DESFire MF3 IC D40

[69] National Institute of Standards and Technology (2001): Federal Information Processing Standards Publication 197 - Announcing the Advanced Encryption Standard (AES), http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[70] NXP Semiconductors (2009): MF3ICD81, MF3ICD41, MF3ICD21 - MIFARE DESFire EV1 contactless multi-application IC, http://www.acs.com.hk/drivers/eng/TDS_DESF_EV1.pdf

[71] NXP Semiconductors (2008): MIFARE DESFire EV1, http://www.nxp.com/acrobat_download2/literature/9397/75016504.pdf

[72] Conty, R. (2011): libnfc - ChangeLog, http://www.libnfc.org/documentation/changelog#v100

[73] o. V. (2011): libnfc - People, https://code.google.com/p/libnfc/people/list

[74] o. V. (2009): libnfc - license, http://www.libnfc.org/license

[75] Conty, R. (2011): libnfc.org - Compatible NFC devices, http://www.libnfc.org/documentation/hardware/compatibility

[76] Conty, R. (2010): libnfc.org - Hardware - PN53X-Chip, http://www.libnfc.org/documentation/hardware/pn53x-chip

[77] Conty, R. (2011): libnfc.org - Feature matrix (per chip), http://www.libnfc.org/documentation/hardware/features

[78] o. V. (2011): nfc-tools development site, https://code.google.com/p/nfc-tools/

[79] Conty, R. (2010): nfc-tools - pam_nfc, https://code.google.com/p/nfc-tools/wiki/pam_nfc

[80] Conty, R. (2010): nfc-tools - mfoc, https://code.google.com/p/nfc-tools/wiki/mfoc

[81] Conty, R. (2011): nfc-tools - libfreefare, https://code.google.com/p/nfc-tools/wiki/libfreefare

[82] Python Software Foundation (2011): ctypes - A foreign function library for Python in Python v2.7.2 documentation, http://docs.python.org/library/ctypes.html

[83] o. V. (2011): Simplified Wrapper and Interface Generator, http://www.swig.org/

[84] o. V. (o. J.): LEGIC_Medien_Uebersicht.pdf, http://www.evis.ch/fileadmin/content/download/Legic_Merkbl_tter/LEGIC_Medien_Uebersicht.pdf

[85] Liechti, C. (2010): pySerial v2.6 documentation, http://pyserial.sourceforge.net/

Abkürzungsverzeichnis

AES Advanced Encryption Standard

AG Aktiengesellschaft

AID Application Identifier

API Application Programming Interface

ATQA Answer to Request

ATS Answer to Select

BSI Bundesamt für Sicherheit in der Informationstechnik

CC Common Criteria

CRC Cyclic Redundancy Check

DES Data Encryption Standard

EAL Evaluation Assurance Level

EEPROM Electrically Erasable Programmable Read-Only Memory

GAM General Autorisation Media

IAM Identification Autorisation Media

IC Integrated Circuit

ICC Integrated Circuit Card

IMSI International Mobile Subscriber Identity

ISO International Organization for Standardization

LFSR Linear Feedback Shift Register

MFOC Mifare Classic Offline Cracker

MTSC Master-Token System Control

NFC Near Field Communication

PC Personal Computer

PIN Personal Identification Number

RFID Radio Frequency Identification

RAM Random-Access Memory

ROM Read-Only Memory

SAK Select Acknowledge

SAM System Autorisation Media

SIM Subscriber Identity Module

SWIG Simplified Wrapper and Interface Generator

UID Unique Identifier

USB Universal Serial Bus

XAM Extended Autorisation Media

XOR Exclusive Or

Abbildungsverzeichnis

Listingverzeichnis