RFID – Sicherheit

Ausarbeitung über praktische Tests mit MIFARE Classic
und den Bau eines RFID-Zappers

von Gerhard Klostermeier
Studiengang: IT-Sicherheit
Vorlesung: Sichere Hardware
November 2011

Inhaltsverzeichnis

1 Einführung

Im Rahmen der Vorlesung „Sichere Hardware“ des Studiengangs „IT-Sicherheit“ der Hochschule Aalen, wurde ein RFID (Radio-Frequency IDentification) Projekt definiert. Ziel dieses Projektes war es, sich mit der RFID-Technik und deren Sicherheit möglichst praktisch auseinander zu setzen. Dabei entstanden drei Schwerpunkte:

- Theoretisches Wissen aneignen.
- Angriffsanalyse für ein real existierendes, autonomes, RFID basiertes System.
- Angriff auf die Verfügbarkeit von RFID-Chips mit einem RFID-Zapper.

Die Arbeitsergebnisse aller drei Themenbereiche werden im Folgenden vorgestellt.

Verweislegende:
Dokument interner Verweis
Externer Verweis (Internet-Link)

1.1 Geschichte

Die Entstehung der RFID-Technik ist wie viele technische Entwicklungen auf das Militär zurückzuführen. Ende des Zweiten Weltkrieges wurden Flugzeuge mit Transpondern ausgestattet, sodass Bodenstationen identifizieren konnten, ob es sich um Freund oder Feind handelt. Als weiterer grundlegender Meilenstein gilt das Paper von Harry Stockman, welches er 1948 unter dem Namen "Communication by Means of Reflected Power" veröffentlichte. In den 60ern wurden weitere proprietäre Systeme entwickelt. 1975 wurde der erste passive Transponder in einem IEEE-Aufsatz vorgestellt. Zuvor waren alle RFID-Technologien aktiv gewesen, d.h. sie benötigten eine eigene Energiequelle und sendeten mit dieser aktiv Signale aus. Während der 70er kamen simple RFID Tags auf den Markt, die beispielsweise in der Warenüberwachung (engl. Electronic Article Surveillance, EAS) oder später verstärkt bei der Tiermarkierung zum Einsatz kamen. Durch die Einführung RFID basierter Mautsysteme in amerikanischen Bundesstaaten, sowie Norwegen wurde die Technik in den 80ern und 90ern nun auch von staatlicher Seite stark gefördert. Die großen Fortschritte ließen zahlreiche andere Anwendungen folgen, wie elektronische Schlösser, Zutrittskontrollen, bargeldloses Zahlen, Skipässe, Tankkarten, elektronische Wegfahrsperren usw. Heute sind durch die Miniaturisierung der Technik, sowie der Entwicklung von temperaturunempfindlichen Tags, der RFID-Technologie kaum noch Grenzen gesetzt. [vergl. Wikipedia]

Der zurzeit (2011) kleinste RFID-Chip ist 0.05mm × 0.05mm groß.[1]
Heute (2011) kostet ein durch Mengenrabatt billiger RFID-Chip ca. 5 cent.[2]

1.2 Verbreitung

RFID ist eine Technologie, die aus dem heutigen Alltag nicht mehr wegzudenken wäre. Im Zeitraum von 1944 bis 2005 wurden laut IDTechEx 2,397 Milliarden RFID-Chips verkauft.[3] Diese Zahl setzt sich aus folgenden Bereichen zusammen:

Branche	Kumulierte Anzahl (in Mio.)
Transport/Automotive	1000
Finanzen/Sicherheit	670
Handel/Konsumgüter	230
Freizeit	100
Wäschereien	75
Bibliotheken	70
Fertigung	50
Tiere/Landwirtschaft	45
Gesundheitswesen	40
Flugverkehr	25
Logistik/Post	10
Militär	2
Sonstige	80
Total	**2397**

Tabelle 1: Kosten pro Bereich nach IDTechEx

Die Anwendungsgebiete zeigen deutlich, dass es nur noch wenige betriebliche Bereiche gibt, die ohne RFID arbeiten. Aber auch ausgefallenere Ideen, wie sich RFID-Chips unter die Haut zu implantieren, finden schon in der Realität Anwendung. Es scheint also kaum Grenzen zu geben. Oft auch auf Kosten des Datenschutzes. So gibt es beispielsweise Vereinigungen (wie z.B. den FoeBuD e.V.), die Kampagnen gegen zu bedenkenlosen Umgang mit RFID führen.

[vergl. Wikipedia]

Abbildung 1: Logo der "Stop RFID" Kampagne des FoeBuD e.V.

1.3 Was ist Proxmark3?

Proxmark3 ist ein Board in Kreditkartengröße, welches von Jonathan Westhues entwickelt und unter der GPL (GNU General Public License) veröffentlicht wurde. Es kann mit nahezu jedem RFID-Tag zwischen 125kHz (Low Frequency) bis 13.56MHz (High Frequency) arbeiten. Gängige Funktionen sind hierbei als Reader zu agieren, Kommunikationen zwischen Tag und einem anderen Reader zu belauschen, Tags zu beschreiben und mittels Emulation sich selbst als Tag auszugeben. Durch die GPL, welche es jedem möglich macht sich selbst so ein Board zu bauen oder Software dafür zu entwickeln, wird der Funktionsumfang jedoch immer wieder erweitert. So gehört beispielsweise das Knacken der Verschlüsselung eines MIFARE Classic 1K Chips, das anschließende Datenauslesen, sowie das anschließend mögliche Erstellen eines Duplikates inzwischen zum Funktionsumfang.

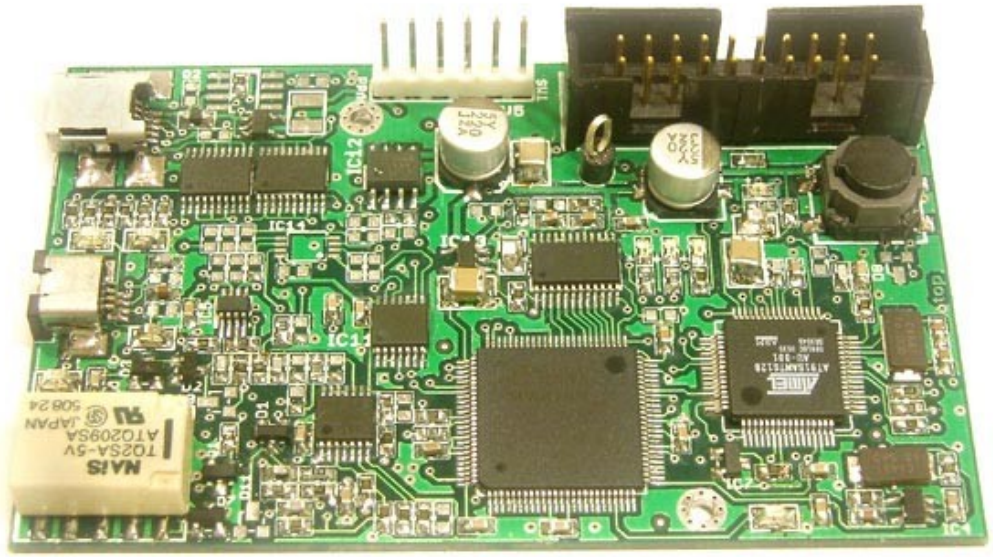

Abbildung 2: Das Proxmark3 Board

Dokumentation und weitere Informationen können den Webseiten unter Weblinks entnommen werden.

2 Technik

Technisch unterscheiden sich RFID-Tags zunächst einmal in Übertragungsfrequenz und Quelle der Betriebsspannung. Gemeinsam haben sie jedoch, dass jeder Transponder eine Antenne, einen analogen sowie einen digitalen Schaltkreis hat.

Frequenzen: Die Übertragungsfrequenzen teilen sich im europäischen Raum wie folgend auf: LF (Low Frequency) 125-134 kHz; HF (High Frequency) 13,56 MHz; UHF (Ultra High Frequency): 865-869 MHz; SHF (Super High Frequency): 2,45 GHz und 5,8 GHz.

Betriebsart: Die Art wie die RFID-Chips mit Strom versorgt werden, ist in zwei Klassen aufzuteilen: Passiv und Aktiv. Passive Tags werden von dem elektromagnetischen Wechselfeld des Readers versorgt, indem sie mittels einer Induktionsspule Strom erzeugen. Bei aktiven Transpondern wird der Microchip mit einer Stromversorgung (Batterie) betrieben. Die Signalübertragung wird aber weiterhin ohne extra Strom realisiert. Deshalb spricht die Literatur hier auch oft von „semi-aktiven“ Transpondern.

Übertragung: Die Datenübertragung wird in verschiedenen Frequenzbereichen unterschiedlich realisiert. Bei einem gängigen HF wird das vom Reader ausgestrahlte Wechselfeld mittels Kurzschließen beeinflusst, sodass dieser durch Messung eine Antwort daraus interpretieren kann. Dies nennt man Lastmodulation. UHF-Tags hingegen nutzen modulierte Rückstreuung. Damit sind sie fähig, viel mehr Daten in einer Passage zu übertragen.

[vergl. Wikipedia]

2.1 MIFARE Classic

Im Rahmen dieses Projektes wurde besonders die MIFARE Classic Technik genutzt. Um folgende Kapitel nachvollziehen zu können, wird hier der Aufbau einer MIFARE Classic 1K Chipkarte erklärt.

Der Speicher einer MIFARE Classic 1K Karte ist in 16 Sektoren unterteilt. Jeder davon besteht aus 4 Blöcken à 16 Byte (Siehe Tabelle).

Kartentyp	Sektoren	Blöcke pro Sektor	Nutzdatenbytes
MIFARE Classic 1K	16	4	768 Bytes 720 Bytes (Sektor 0 = MAD)
Zum Vergleich: MIFARE Classic 4K	32 + 8	4 (Sektor 0 - 31) 16 (Sektor 32 - 39)	3456 Bytes 3360 Bytes (Sektor 0 + 16 = MAD)

Tabelle 2: Aufbau von MIFARE Classic (Quelle: Wikipedia)

In Sektor 0 ist zunächst einmal Block 0 zu betrachten. Die ersten 4 Byte stellen die UID (Unique IDentification number) dar. Da inzwischen so viele MIFARE-Chips verkauft wurden, ist diese Bezeichnung etwas irreführend, denn die Nummern sind nicht mehr einzigartig (unique). Die restlichen Bytes sind herstellerspezifische Informationen. Eine weitere Besonderheit ist, dass der Block nur gelesen werden kann.

Der letzte Block jedes Sektors wird Sector Trailer genannt. Mit ihm wird der für jeden Sektor individuelle Lese- und Schreibzugriff geregelt. Er ist in 3 Segmente unterteilt. Die ersten 6 Byte sind der Schlüssel A. Es folgen 4 Byte Zugriffsrechte (Access Conditions) und wieder 6 Byte für den Schlüssel B. In den Access Conditions ist genau geregelt, was in welchem Block des Sektors mit welchem Schlüssel gemacht werden darf. Außerdem ist in den Zugriffsrechten festgelegt, ob diese selbst verändert werden dürfen, oder ob beispielsweise Schlüssel geändert werden können. Die Datenblock betreffenden Rechte sind in "read", "write", "incr" und "decr, transfer, restore" unterteilt. (Eine gängige Konfiguration ist es, den Schlüssel A zum Lesen und den Schlüssel B zum Schreiben zu verwenden.)

[vergl. Wikipedia]

Die im vorigen Absatz genannten "incr" und "decr, transfer, restore" Operationen können nur auf bestimmt formatierten Blöcken ausgeführt werden. Diese Blöcke werden "Value Block" genannt. Ihr Aufbau ist wie folgend:

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	value				value				value				adr	$\overline{\text{adr}}$	adr	$\overline{\text{adr}}$

Abbildung 3: Aufbau eines Value Blocks (Quelle: MIFARE data sheet (nxp.com))

Value: Ein 4 Byte Wert der zur Sicherheit dreimal gespeichert wird (davon einmal invertiert). Dieser kann inkrementiert oder dekrementiert werden.

Address: Ein 1 Byte Wert welcher viermal gespeichert wird. Er wird intern nicht interpretiert.

Achtung: Die Abbildung zeigt einen solchen Value Block in Little-Endian Schreibweise. Um den Wert wie in üblichen Zahlensystemen mit absteigendem Stellenwert zu verarbeiten, muss er in Big-Endian Notation konvertiert werden.

Beispiel: Value Block: 0x8F00000070FFFFFF8F00000000FF00FF.
Der Abgespeicherte Wert ist: 0x0000008F = $143_{(D)}$

[vergl. MIFARE data sheet]

3 Angriff auf ein autonomes MIFARE-System

Dieser Abschnitt betrachtet ein reales autonomes System, bei dem die MIFARE-Technologie zum Einsatz kommt. Konkret werden RFID-Tags verwendet, um den Zugang zu Räumen zu sichern, Geräte zu bedienen (z.B. Kopierer) und Einkäufe zu tätigen. Dabei wird davon ausgegangen, dass der Angreifer im Besitz eines solchen Tags ist (legal oder illegal).

3.1 Analyse des Systems

Der erste Blick gilt einer groben Analyse der Einrichtung. Ziel ist es, eine Übersicht über die Komponenten und deren Zusammenspiel zu bekommen. Des Weiteren sind auf Dinge zu achten, die später bei dem Daten extrahieren und analysieren hilfreich sein können.

- **Allgemein:** Ein Scan einer Karte gibt Auskunft darüber, ob es sich um ein MIFARE Classic 1K Tag handelt - die vermutlich für das ganze System eingesetzte Technologie.
- **Zugang:** Vor bestimmten Räumen sind RFID-Reader zu finden. Mit einem für diesen Raum freigeschalteten Tag lässt sich die Tür öffnen. Wenn ein nicht freigeschalteter Mitarbeiter Zugang zum Raum haben will, so muss er dies bekannt geben. Der Zugang wird dann freigeschaltet ohne dass der Tag manipuliert wird. Die logische Konsequenz daraus ist, dass der RFID-Reader offensichtlich eine auf dem Tag gespeicherte ID (z.B. Personalnummer) gegen eine ihm bekannte Datenbank prüft. Wenn es möglich ist diese ID zu extrahieren oder zu manipulieren, kann ein Angreifer sich möglicherweise Zugang zu einem solchen Raum verschaffen.
- **Einkaufen:** Mit dem Tag ist es möglich an Automaten oder einer Caféteria zu bezahlen. Zuvor wurde hierfür an einem anderen Automat mit dem Tag das Konto aufgeladen. An den Automaten ist zu beobachten, dass jeder eine Netzwerkverbindung hat (sichtbares Kabel). Es folgen daraus drei unterschiedliche Möglichkeiten:
    - Das Guthaben ist rein virtuell auf dem RFID-Tag gespeichert und die Netzwerkverbindung für den Einkauf irrelevant. Folge: Wenn das Konto auf dem Tag manipuliert werden kann, ist es möglich Geld zu erschaffen.
    - Das Guthaben wurde beim Aufladevorgang in einer Datenbank hinterlegt. Bei einem Bezahlvorgang wird die ID (Personalnummer) in der Datenbank nachgeschlagen und das Geld vom dortigen Konto abgebucht. Folge: Wenn die ID zu extrahieren oder zu manipulieren ist, kann auf ein Konto eines Mitarbeiters eingekauft werden.
    - Das Guthaben ist auf dem Tag und in einer Datenbank hinterlegt worden. Folge: Nur wenn ein Tag komplett (Konto und ID) dupliziert werden kann, so ist es möglich auf das Konto eines Mitarbeiters einzukaufen. Allerdings nur bis dieser selbst etwas einkauft. Dann fällt die Differenz zwischen Datenbankkonto und dem noch unbelasteten Transponder-Konto des Mitarbeiters auf. Den Täter aufgrund von Logfiles zu finden bleibt aber immer noch unmöglich.

- **Gerätenutzung (z.B. Kopierer):** Es ist möglich Dokumente zu kopieren oder auszudrucken. Auch hier kommt der RFID-Tag zum Einsatz. Ähnlich wie beim Einkaufen gibt es für dieses System auch ein Konto, welches eine bestimmte Anzahl an Ausdrucken ermöglicht. Ebenfalls wie beim Einkauf hat auch jeder RFID-Reader zur Drucküberwachung ein Netzwerkkabel. Es gelten also die selben drei Möglichkeiten wie beim Einkauf.

3.2 MIFARE-Verschlüsselung brechen

Das Brechen der MIFARE-Verschlüsselung ist seit 2008 möglich. MIFARE Classic Chips verschlüsseln auf Basis der proprietären Stromchiffre Crypto-1. Diesen Algorithmus konnten Forscher des Chaos Computer Clubs und der University of Virginia rekonstruieren.[4] [5] Dies geschah durch das schichtweise Abschleifen von MIFARE Chips und die anschließende Untersuchung unter dem Rasterelektronenmikroskop. Aufgefallen ist dabei auch, dass der Pseudozufallszahlengenerator der Tags, sowie der Reader fehlerhaft sind.[6] Durch diese Erkenntnis wurde es möglich, Klone eines RFID-Tags zu generieren. Dazu musste aber der geheime Schlüssel aus einem Reader gewonnen werden. Später jedoch (2009) entdeckte Nicolas T. Courtois weitere Probleme, die diesen Schritt überflüssig machen.[7]

Es wurden eigene praktische Tests durchgeführt, um das Brechen der Verschlüsselung nachzuvollziehen. Zum Einsatz kam dabei nur ein Computer, das Proxmark3 Board und die zu knackende MIFARE Classic 1K Karte. Es gelang alle 16 Schlüssel der Sektoren in ca. 30 Minuten zu extrahieren. (Im englischen Wikipedia wird erwähnt, dass mit der richtigen Hardware dieser Vorgang in 10 Sekunden möglich sei.)

3.2.1 "Dark Side Attack"

Die Gewinnung aller Schlüssel für ein MIFARE Classic 1K Tag (16 Stück) läuft in zwei Schritten ab. Zu Beginn wird mittels der so genannten "Dark Side Attack" der Schlüssel des ersten Sektors geknackt. Dazu wurde von der Community des Proxmark3 diese Angriffsmethode implementiert. Eigene Praxistests zeigten, dass dieser Angriff ca. 10 Minuten dauert. Dies kann laut Angaben der Community aber variieren.

3.2.2 "Nested Authentication Attack"

Nachdem der Schlüssel des Sektors 1 bekannt ist, wird nun mit der "Nested Authentication Attack" fortgefahren.[8] Diese kann sobald ein Schlüssel bekannt ist, alle anderen Schlüssel des RFID-Tags knacken. Auch diese Attacke wurde von der Community für das Proxmark3 Board implementiert. So ist es möglich, nach einer einfachen Befehlseingabe alle Schlüssel zu gewinnen. Praxistests zeigten, dass dieser Vorgang in ca. 10-30 Minuten abgeschlossen ist.

3.3 Daten extrahieren und analysieren

Wenn nun die Schlüssel aller Sektoren erlangt wurden, so können die Daten auf übliche Weise gelesen werden. Das Resultat ist ein sogenanntes "Hexdump", eine Datei in der alle Informationen (Sektoren mit ihren Blöcken) in Hexadezimal einsehbar sind. Nun geht es darum, die Daten ihren Funktionen zuzuordnen. Aus der Analyse des Systems geht hervor, dass auf der Chipkarte vermutlich drei Datensätze zu finden sind. Diese mühsame Arbeit des Zuordnens fällt in den Bereich des „Reverse Engineering“.

Achtung: Im Folgenden werden immer wieder Erkenntnisse an konkreten Speicherauszügen eines MIFARE Transponders erklärt. Dabei wurden alle Daten abgeändert, um das System nicht zu gefährden. Des weiteren wurden alle Schlüssel in den Sector Trailern durch Nullen ersetzt. Betrugsmöglichkeiten wie Identitätsdiebstahl sind damit ausgeschlossen.

Beispiel für den Inhalt eines MIFARE Classic 1K Tags:

```
s00
5746EE09872804002328287F00030104
050203F405000000090C0B0C0D0E8089
070809200F1201B17B239F1E07660862
00000000000063C78900000000000000
s01
22040000DDFBFFFF2204000000FF00FF
22040000DDFBFFFF2204000000FF00FF
000000000000000000000000000000FF
00000000000040F78B00000000000000
s02
2148050707DB0A0E000001C4000000C8
1B410B0707DB0F3200000000000086D5
5E9E005E9E005B9E005F9E000000003F
00000000000072D788000000000000000
s03
3104201030000000030032050236 00C3
3100000004525600004 0C1000067101A
01092013C4FAEE90E5000000000000E9
00000000000070F788000000000000000
s04
313635323534303030303012026608C0
313635323534303030303012026608C0
000000000000000000000000000000000
00000000000070F788000000000000000
s05
303030343532353600003020000000E3
000000000000000000000000000000000
000000000000000000000000000000000
00000000000070F788000000000000000
s06
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
000000000000787788000000000000000
s07
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
000000000000787788000000000000000
s08
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
000000000000787788000000000000000
s09
00000000FFFFFFFF0000000000FF00FF
00000000FFFFFFFF0000000000FF00FF
000000000000000000000000000000FF
00000000000040F78B00000000000000
s10
1302060807D90B0C000000000000003F
1302060807D90B0C000000000000003F
000000000000000000000000000000FF
00000000000072D788000000000000000
s11
00000000FFFFFFFF0000000000FF00FF
00000000FFFFFFFF0000000000FF00FF
000000000000000000000000000000FF
00000000000040F78B00000000000000
s12
1302060807D90B0C000000000000003F
1302060807D90B0C000000000000003F
000000000000000000000000000000FF
00000000000072D788000000000000000
s13
F10000000EFFFFFFF100000000FF00FF
F10000000EFFFFFFF100000000FF00FF
000000000000000000000000000000FF
00000000000040F78B00000000000000
s14
1A01150307DB0B1B0000012C142000F3
110B0C0707DB0F2E0000000000001A4B
000000000000000000000000000000FF
00000000000072D788000000000000000
s15
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
000000000000787788000000000000000
```

3.3.1 Zugang

Durch die Analyse des Systems wird vermutet, dass der Zugang über die Personalnummer abläuft. Eine solche Nummer ist auf der Karte auch selbst abgedruckt. Beim „Reverse Engineering“ besteht zumeist der erste Schritt darin, Daten in ASCII codiert zu finden.

Beispiel:
Die Personalnummer lautet: 45256.
In Sektor 04 wird man fündig:

```
s04
313635323534303030303012026608C0
313635323534303030303012026608C0
00000000000000000000000000000000
00000000000070F78800000000000000
```

Die ASCII-Decodierung der roten Daten ist: 65254. Dies ist die Personalnummer – lediglich invertiert. In Sektor 05 wird man abermals fündig:

```
s05
303030343532353600003020000000E3
00000000000000000000000000000000
00000000000000000000000000000000
00000000000070F78800000000000000
```

Diesmal entspricht der Fund genau 45256 (mit drei führenden Nullen). Nach genauer weiterer Betrachtung fällt die Nummer ein drittes mal ins Auge:

```
s03
310420103000000003003205023600C3
310000000452560000040C1000067101A
01092013C4FAEE90E5000000000000E9
00000000000070F78800000000000000
```

Hier wurden die Hexadezimal-Daten nicht in ASCII codiert, sondern so, dass sie für Menschen direkt ablesbar sind. Eine eher untypische Methode.

3.3.2 Einkaufen

Ähnlich wie beim Zugang wird auch hier so vorgegangen, dass bekannte Werte gesucht werden. Des weiteren werden zusätzliche Umstände berücksichtigt. Wenn das Konto mit dem Guthaben tatsächlich auf dem RFID-Tag gespeichert sein sollte, so ist es wahrscheinlich, dass eben dieses Guthaben in einem Value Block auftaucht. Weiterhin ist es üblich, Kontostände in Ganzzahl-Arithmetik zu implementieren. Das Guthaben sollte also in Cent auffindbar sein.

Angenommen es sind 10,58€ auf dem Transponder gespeichert. Dieser Betrag liegt aber als Cent vor, also: 1058 Cent. In Hexadezimal sind das 0x422 Cent. Value Blocks speichern jedoch in Little-Endian, d.h. die gesuchten Zeichen sind „2204“.

```
s01
220400000DDFBFFFF2204000000FF00FF
220400000DDFBFFFF2204000000FF00FF
000000000000000000000000000000FF
00000000000040F78B00000000000000
```

Sofort ist auch die Struktur eines Value Blocks wieder zu erkennen (siehe MIFARE Classic). Dieser Wert wird redundant gespeichert.

Eine weitere interessante Untersuchungsmethode bei variablen Speicherständen wie diesem Konto ist es, den selben Transponder vor und nach einem Einkauf auszulesen. Durch einen anschließenden Vergleich ist festzustellen, welche Bits sich im Speicher geändert haben. Wenn dieser Test öfters an verschiedenen Einkaufsmöglichkeiten wiederholt und die Karte zwischendurch mit neuem Guthaben aufgeladen wird, fällt der Folge-Sektor eines Kontos ins Auge.

```
s02
2148050707DB0A0E000001C4000000C8
1B410B0707DB0F3200000000000086D5
5E9E005E9E005B9E005F9E000000003F
00000000000072D78800000000000000
```

Durch die verschieden Einkäufe zu verschiedenen Zeitpunkten ist der grüne und der blaue Teil auffällig geworden. Durch einzelne Betrachtungen fällt auf, dass Grün als „05-07-07DB“ zu lesen ist. In Dezimal bedeutet das „05-07-2011“, dies entspricht dem Einkaufsdatum. Blau ist die dazu passende Einkaufsuhrzeit, „0A-0E“ = „10:14“. Die verschiedenen Einkaufsorte heben den roten Bereich hervor. Tests ergaben, dass es sich hierbei um die Kassen- / Automatenkennung handelt.

Der erste Block repräsentiert den Zeitpunkt des letzten Aufladevorgangs, der zweite den des letzten Einkaufs.

Jedes Mal ändert sich auch das türkis markierte Byte mit. Dieses Byte ist auch in vielen anderen Blöcken verschiedener Sektoren ähnlich zu sehen. Die Anordnung am Ende eines Blocks, sowie die regelmäßigen Änderungen die bei Einkäufen auftreten, lassen Rückschlüsse zu, dass dies eine Prüfsumme ist. An dieser Stelle wäre ein kryptographischer MAC (Message Authentication Code), der die Datenintegrität garantieren soll zu erwarten. Jedoch zeigten viele simple Ansätze irgendwann Erfolg. So gelang es den Algorithmus zur Berechnung der Prüfsumme zu rekonstruieren. Er lautet:

```
B15 = 0xFF - (B0 XOR B1 XOR B2 XOR B3 XOR B4 XOR B5 XOR B6 XOR B7 XOR B8 XOR B9
      XOR B10 XOR B11 XOR B12 XOR B13 XOR B14)
```

Es werden demnach alle Bytes (bis auf das Letzte) eines Blocks mit XOR verknüpft und anschließend von 0xFF abgezogen.

Der Block 01 mit der selben Einfärbung ändert sich analog zum Block 00, wenn das Kartenguthaben neu aufgeladen wurde.

3.3.3 Gerätenutzung (z.B. Kopierer)

Wie in der Analyse des Systems vermutet, wird z.B. das Kopierkonto gleich gehandhabt wie das Einkaufskonto. So wurden hier mit den gleichen Methoden wie im Kapitel Einkaufen die Informationen extrahiert.

```
s13
F10000000EFFFFFFF100000000FF00FF
F10000000EFFFFFFF100000000FF00FF
000000000000000000000000000000FF
00000000000040F78B00000000000000
s14
1A01150307DB0B1B0000012C142000F3
110B0C0707DB0F2E0000000000001A4B
000000000000000000000000000000FF
00000000000072D78800000000000000
```

Der Aufbau ist identisch mit dem des Einkaufskontos.

3.3.4 Weitere gefundene Informationen

Sektor 09/10, sowie Sektor 11/12 weisen abermals den Aufbau eines Kontos auf.

```
s09
00000000FFFFFFFF0000000000FF00FF
00000000FFFFFFFF0000000000FF00FF
0000000000000000000000000000000FF
00000000000040F78B00000000000000
s10
1302060807D90B0C000000000000003F
1302060807D90B0C000000000000003F
0000000000000000000000000000000FF
00000000000072D78800000000000000


s11
00000000FFFFFFFF0000000000FF00FF
00000000FFFFFFFF0000000000FF00FF
0000000000000000000000000000000FF
00000000000040F78B00000000000000
s12
1302060807D90B0C000000000000003F
1302060807D90B0C000000000000003F
0000000000000000000000000000000FF
00000000000072D78800000000000000
```

Die Konten scheinen ungenutzt aber initialisiert zu sein. Der genaue Aufbau wird im Kapitel Einkaufen beschrieben.

Des weiteren ist noch im Sektor 03 das Kartengültigkeitsdatum direkt ablesbar.

```
s03
01092013300000000300320502360 0C3
3100000004525600004 0C1000067101A
31042010C4FAEE90E5000000000000E9
00000000000070F78800000000000000
```

Im Beispiel ist der Tag vom 31.04.2010 bis zum 01.09.2013 gültig.

3.4 Mögliche Folgen

Dadurch dass die Schlüssel eines MIFARE Classic Transponders einfach durch Geräte wie den Proxmark3 geknackt werden können, ist es überhaupt möglich gespeicherte Daten zu manipulieren. Es könnte also jedes beliebige Byte geändert werden, jedoch mit Einschränkungen.

Beispielsweise kann man die Personalnummer ändern, um zu anderen Räumen Zugang zu erhalten. Da die Sektoren und Blöcke in denen sie auftritt nicht als „read only“ markiert sind, reicht ein einfaches Abändern und das Neubilden der Prüfsumme. Der Betrug könnte nur auffallen, wenn der Zugangs-Kontroll-Reader zusätzlich zur Personalnummer auch die UID des RFID-Tags prüft. Diese mögliche Sicherheitsbarriere kann nur mit spezieller Hardware, wie dem Proxmark3 überwunden werden. Er kann MIFARE Classic 1K Transponder emulieren und dabei einen beliebigen Block 00 im Sektor 00 zugewiesen bekommen.

Für die Konten gelten ebenfalls die Vorgehensmöglichkeiten wie für den Zugang im vorigen Absatz beschrieben.

Um einen „100% Klon“ mit dem Proxmark3 zu realisieren wurde ein Skript entwickelt, welches mit einer Klon-Vorlage und einer Schlüsseldatei das Gerät so initialisiert, dass es den zur Vorlage identischen Tag darstellt. Die Vorlagedatei kann mit dem erweiterten Demo-Programm von Omnikey (siehe Kapitel Demo-Software Abänderungen) erstellt werden. Die Schlüsseldatei enthält die Ausgabe der "Nested Authentication Attack".

Abschließend sind also folgende Punkte zu erkennen:

- Die Herstellung einer geklonten Karte ist möglich.
- Wenn dabei die UID auch der Vorlage entsprechen soll, so ist besondere Hardware (Proxmark3) nötig.
- Eine Karte kann beliebige manipuliert werden.
  (Personalnummer ändern, Kontostand ändern, usw.)
- Manipulierte Karten können möglicherweise auffallen (z.B. beim Einkaufen), da wahrscheinlich Daten mit einer Datenbank abgeglichen werden.

4 Angriff auf die Verfügbarkeit (Bau eines RFID-Zappers)

Durch die hohe Verbreitung der RFID-Tags, ist die Verfügbarkeit umso wichtiger. Unternehmen verlassen sich auf diese Chips, um ihre Lager zu verwalten. Kaufhäuser sichern ihre Ware damit gegen Diebstahl. Sogar der neue Personalausweis ist mit RFID ausgestattet.[9] Was ist aber, wenn man ein solches RFID-Modul zerstören könnte, ohne dass der "Eingriff" sichtbar wäre. Die Lagerautomatik wüsste nicht mehr wohin mit den Gütern, das Kaufhaus kann bestohlen werden und der neue Personalausweis (nPA) ist nur noch eine Plastikkarte, so wie sein Vorgänger. Ein solcher Angriff wird mit einem sogenannten RFID-Zapper realisiert. Um den technischen Vorgang dabei genau nachzuvollziehen, wurde ein solches Gerät gebaut.

4.1 Vorbereitung

Die grundlegende Idee des Aufbaus wurde 2005 auf dem 22c3 vorgestellt und ist im CCCWiki nachzulesen. (Weitere Anleitungen unter Weblinks.) Dabei ist das Prinzip immer gleich. Eine Blitzschaltung eines alten Fotoblitzes oder einer Einwegkamera, wird anstatt mit der Blitzröhre mit einer selbst gemachten Spule verlötet. (Eine Einwegkamera eignet sich besser; siehe Test.) Wenn sich so der voll aufgeladene Kondensator (ca. 300V) der Blitzschaltung über die Spule entlädt, wird ein elektromagnetischer Puls (EMP) erzeugt. Der RFID-Chip besitzt ebenfalls eine Spule als Antenne und zur Stromversorgung. Im Regelfall wird mit dem vom Leser ausgesandten elektromagnetischen Feld Strom induziert, um den Chip zu betreiben. Das elektromagnetischen Feld (EMP) des Zappers ist allerdings zu stark und lässt den Chip durchbrennen, ohne äußerlichen Schaden zu hinterlassen.

4.2 Bau

Achtung: Falls Sie selbst einen Zapper bauen, sollten Sie nicht vergessen dass Teile der Schaltung im aufgeladenen Zustand über 300V haben können. Zum sicheren Entladen sollten die Pins des Kondensators mit einem ca. 100K-Ohm Widerstand verbunden werden. Mit einem Multimeter kann geprüft werden, ob der Kondensator entladen ist. Erst dann ist das Arbeiten an der Schaltung ungefährlich.

Erster Schritt ist, das Gehäuse des Fotoblitzes oder der Einwegkamera zu entfernen. Anstelle der Blitzröhre wird eine Spule mit einem vorgeschalteten Taster gelötet. Die Spule selber ist aus einem ca. 1mm Draht mit ca. 5 Wicklungen in Kreditkartengröße gefertigt. Beim Anlöten der Spule ist zu beachten, dass sie an den vormals linken und rechten Pol der Blitzröhre gehört. Xenon-Blitzröhren haben normalerweise 3 Pins. Der Dritte wird für die Zündung der Gasentladung genutzt. Für den Bau des Zappers ist er aber irrelevant und wird ungenutzt bleiben. Der fertige RFID-Zapper sollte dann wieder in ein isoliertes Gehäuse gebaut werden, um den unbeabsichtigten Kontakt mit der unter hoher Spannung stehenden Schaltung zu vermeiden.

4.3 Test

Getestet wurden zwei RFID-Zapper-Varianten. Der Erste basierte auf einem alten Fotoblitz mit einer Spule aus ca. 2mm dickem Draht. Bei einer Ladung von 300V wurde der RFID-Tag zwar einwandfrei zerstört, allerdings war die Schaltung der hohen Belastung nicht gewachsen, sodass einige Bauteile zu Bruch gingen. Die zweite Variante wurde aus einer Einwegkamera gebaut. Da diese Schaltung wesentlich kleiner, einfacher auszubauen und schon meist mit 1,5V betreibbar ist, ist sie für den Bau eines Zappers besonders gut geeignet. Ebenso wurde die Spule verkleinert, indem ein 1mm Draht zum Einsatz kam. Um die Schaltung nicht zu sehr zu belasten, wurden Tests mit einer Kondensatorspannung von ca. 150V aufwärts durchgeführt. Diese reichten schon aus, um RFID-Tags wie gewollt zu zerstören. Das Problem, dass im geschlossenen RFID-Zapper Gehäuse nicht mehr die Spannung am Kondensator nachgemessen werden kann, ist zu lösen indem eine Einwegkamera mit Blitzkontrollleuchte verwendet wird. Wenn die Leuchte schon leicht zu glimmen beginnt, sollte die nötige Spannung vorhanden sein.

Als "Proof of concept" (PoC) wurde ein Video erstellt, welches frei zum Download steht (siehe Inhalt der beiliegenden CD).

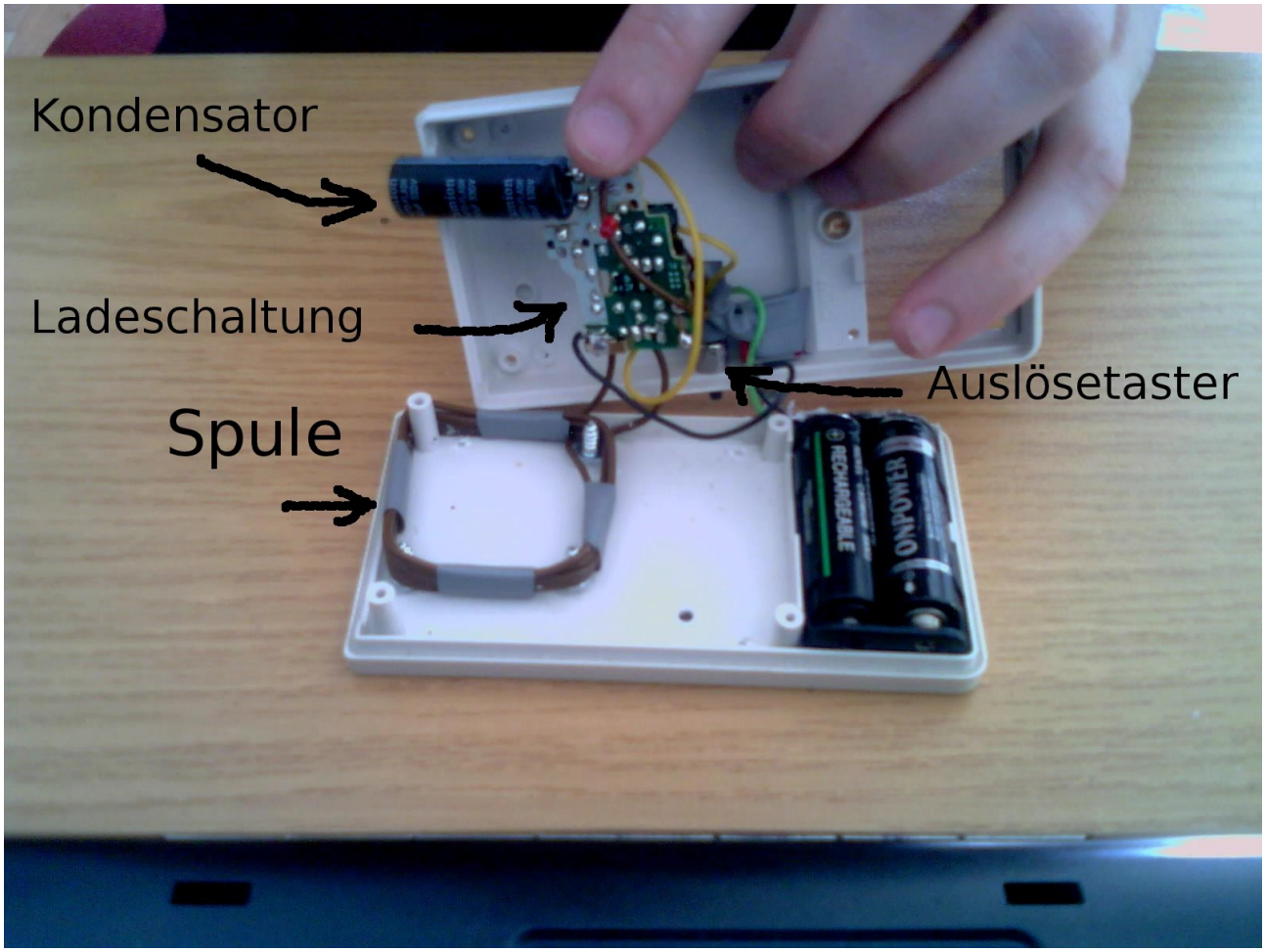

Abbildung 4: Offener RFID-Zapper mit grober Beschriftung

5 Genutzte Hardware

Folgende Hardware kam zum Einsatz, um die Analysen und Angriffe durchzuführen.

5.1 Proxmark3

"The Proxmark III is a device developed by Jonathan Westhues that enables sniffing, reading and cloning of RFID (Radio Frequency Identification) tags." (Quelle: proxmark.org)

Weitere Informationen auch im Kapitel: Was ist Proxmark3?.

5.1.1 Firmware selbst Kompilieren

Achtung: Dieser und der folgende Abschnitt ist inzwischen veraltet. Das Proxmark3-Projekt enthält nun eine Ausführliche Anleitung über das Kompilieren und Flashen des des RFID-Boards.

Es wird dringend empfohlen, den Proxmark3 mit der neuesten Firmware zu betreiben. Da sich die Firmware durch die Community ständig weiterentwickelt, entfallen ansonsten möglicherweise wichtige und interessante Funktionen. Falls das Board noch die original Firmware besitzt, unterscheiden sich die Flash-Operationen. Praktische Hinweise finden Sie unter proxmark.org. (Einige Teile der dortigen Anleitung sind veraltet.)

Installation der benötigten Software unter Debian/Ubuntu:

```
sudo apt-get install subversion build-essential libreadline5 libreadline-dev
libusb-0.1-4 libusb-dev libqt4-dev perl pkg-config ↲
```

Des Weiteren wird ein ARM-Cross-Compiler benötigt. Hierfür bietet sich devkitARM an. Nach dem Downlaod von devkitARM, muss das Paket entpackt werden. Als Speicherort bietet sich das Heimatverzeichnis an. Die Binaries befinden sich demnach dann unter:

```
/home/username/devkitARM/bin
```

Jetzt müssen die devkitARM Binaries in $PATH aufgenommen werden. Es reicht ein simples

```
PATH="$PATH:/home/username/devkitARM/bin" ↲
```

Wenn man dies nicht bei jedem Konsolenstart vor dem Kompilieren neu machen möchte, kann man die `.bashrc` um die vorige Zeile ergänzen.

Der nächste Schritt besteht darin, den Quelltext der Firmware herunterzuladen. Dies passiert über das Versionsverwaltungsystem "Subversion". Hierzu wird folgende Zeile eingegeben:

```
svn co http://proxmark3.googlecode.com/svn/trunk proxmark3-read-only ↲
```

Danach mit `cd proxmark3-read-only` ↲ in das heruntergeladene Verzeichnis wechseln. Bei einem Test stellte sich heraus, dass der Quelltext auf einem Ubuntu 11.04 nicht ohne Warnungen zu kompilieren ist. Die Standardkonfiguration behandelt Compiler-Warnungen aber als Fehler, was zum Abbruch führt. Da die aufgetretenen Warnungen allerdings zu vernachlässigen sind, schafft das Abändern der Konfiguration Abhilfe. In der Datei `common/Makefile.common` ist in der Zeile

```
CFLAGS = -c $(INCLUDE) -Wall -Werror -pedantic -std=gnu99 $(APP_CFLAGS)
```

die Flag `-Werror` zu entfernen.

Nun kann die Firmware kompiliert werden. Dafür gibt man im proxmark3-read-only Verzeichnis `make` ↲ ein.

5.1.2 Proxmark3 Firmware flashen

Falls das Board noch die original Firmware besitzt, unterscheiden sich die Flash-Operationen. Praktische Hinweise finden Sie auf proxmark.org. (Einige Teile der Anleitung sind veraltet.)

Die Eingabe von `make help` ↲ zeigt folgende Flash-Optionen:

```
+ flash-bootrom - Make bootrom and flash it
+ flash-os - Make armsrc and flash os
+ flash-fpga - Make armsrc and flash fpga
+ flash-both - Make armsrc and flash os and fpga image
+ flash-all - Make bootrom and armsrc and flash bootrom, os and fpga image
```

In den meisten Fällen reicht die Option `flash-both`. Der „bootrom“ sollte nicht ständig neu geflasht werden. Es genügt ihn dann zu flashen, wenn es eine Änderung gab, die diesen beeinträchtigt.

5.2 HID Omnikey CR 5321 USB RFID-Reader

Bei diesem Gerät handelt es sich um einen kommerziell vertriebenen RFID-Reader/Writer. Er bietet nicht mehr Funktionen wie das Proxmark3 Board, jedoch ist das Entwickeln von Programmen durch die APIs wesentlich komfortabler. Herunterladbare Demo-Programme erleichtern den Einstieg. Mehr Informationen können der Hersteller-Webseite entnommen werden.

5.2.1 Demo-Software Abänderungen

Im Laufe des Projektes wurde die Demo-Software des Herstellers angepasst. Dabei entstanden folgende Features:

- „Brute Force“ oder „Random Try“ Angriffsmethoden wurden implementiert. (Zum Knacken der Schlüssel ist dies nicht zu empfehlen, da diese Methoden viel zu langsam sind.)
- Wenn alle Schlüssel der Sektoren bekannt sind, können diese auf den Leser übertragen und gespeichert werden.
- Wenn die Schlüssel auf dem Gerät sind, kann mit einem Klick der komplette MIFARE-Classic 1K Speicher ausgelesen und angezeigt werden.
- Wenn die Schlüssel auf dem Gerät sind, kann der Reader in einen „Dauer-Auslesen-Zustand“ versetzt werden. Wenn jetzt ein Tag in die Nähe des RFID-Readers kommt, wird er sofort ausgelesen und die Daten gespeichert. Dieser Vorgang dauert ca. 1,5 Sekunden. Wenn der Transponder während des Auslesens den Empfangsbereich verlassen sollte, werden die Daten bis zu diesem Zeitpunkt gespeichert, so wie der Name der Speicherdatei um ein „-error“ ergänzt.

Inzwischen wurden ähnliche Features durch die Community auch im Proxmark3 implementiert.

Achtung: Es ist zu beachten, dass es sich bei den Änderungen im Demo-Programm lediglich um ein „Proof of concept“ handelt. Das heißt, die Erweiterungen sind undokumentiert, teilweise schlecht geschrieben und können bei falscher Benutzung das Programm zum Absturz bringen.

5.3 Sonstige

- Fotoblitz für den Bau eines RFID-Zappers.
- Einwegkamera für den Bau eines RFID-Zappers.

6 Weblinks

- Community Webseite des Proxmark3
- Proxmark3 Shop
- (Veraltete) Webseite des Proxmark3 Entwicklers
- Anleitung zum Bau eines RFID-Zappers
- Weitere Anleitung zum Bau eines RFID-Zappers

7 Inhalt der beiliegenden CD

Der Inhalt steht auch als 7zip zum Download bereit.
(Die Dokumentation als .odt ist in der Download-Version nicht enthalten.)

Dokumentation:

- Diese Dokumentation als LibreOffice-Dokument (.odt) und als PDF.

Unterlagen:

- „A Practical Attack on the MIFARE Classic“, Veröffentlichung der Universität Radboud als PDF.
- „The Dark Side of Security by Obscurity and Cloning MiFare Classic Rail and Building Passes Anywhere, Anytime“ als PDF.
- „NXP MF1S503x Product data sheet“ als PDF.
- 22C3 Wiki-Artikel über das dort vorgestellte RFID-Zapper-Projekt als PDF.
- Proxmark3 Dokumentation in verschiedenen Dateiformaten.

Programme:

- „ContactLessDemoVB“, eine erweiterte Version der Hersteller Demo-Software als Setup-Paket und als Quelltext (Visual Basic 6).
  Siehe Demo-Software Abänderungen.
- „clone.sh“, ein Bash / Expect Skript welches den Proxmark3 als perfekten Klon initialisiert. Siehe Mögliche Folgen.

RFID-Zapper:

- Bilder vom Bau des RFID-Zappers (.jpg).
- Ein „Proof of concept“ Video, dass den RFID-Zapper beim Einsatz zeigt.
  (Container: mp4, Videocodec: H.264, Audiospur: keine.)

[1] Hitachi's RFID powder freaks us the heck out", Engadget, 2010-04-24. (http://www.engadget.com/2007/02/14/hitachis-rfid-powder-freaks-us-the-heck-out)
[2] Roberti, Mark (2006-05-06). "A 5-Cent Breakthrough". RFID Journal. 2007-01-26. (http://www.rfidjournal.com/article/articleview/2295/1/128/)
[3] RFID tag sales in 2005 – how many and where, IDTechEx, 21. Dezember 2005. (http://www.idtechex.com/research/articles/rfid_tag_sales_in_2005_how_many_and_where_00000398.asp)
[4] Mifare – Little Security, Despite Obscurity, Vortrag auf dem 24C3. (http://events.ccc.de/congress/2007/Fahrplan/events/2378.en.html)
[5] Verschlüsselung eines führenden Bezahlkartensystems geknackt, Artikel im Heft 08/2008 der c't (http://www.heise.de/kiosk/archiv/ct/2008/8/80_Verschluesselung-eines-fuehrenden-Bezahlkartensystems-geknackt)
[6] A Practical Attack on the MIFARE Classic, Veröffentlichung der Universität Radboud (http://www.cs.ru.nl/F.Garcia/publications/Attack.MIFARE.pdf)
[7] The Dark Side of Security by Obscurity and Cloning MiFare Classic Rail and Building Passes Anywhere, Anytime, 2009-05-04 (http://eprint.iacr.org/2009/137)
[8] Digital Security Group, "Security Flaw in Mifare Classic", Radboud University Nijmegen, 2008-03-12. (http://www.ru.nl/ds/research/rfid/)
[9] Informationsseite des Bundesministeriums des Innern (http://www.personalausweisportal.de/)