„Betrachtung der MIFARE-Sicherheit anhand von Praxisbeispielen”

Studienarbeit im Fach Informatik

vorgelegt von

Johannes Schlumberger

geb. am 25.01.1983

Angefertigt am

Lehrstuhl für Informatik 4
(Verteilte Systeme und Betriebssysteme)

Friedrich-Alexander-Universität Erlangen-Nürnberg

Betreuer: **Prof. Dr. W. Schröder-Preikschat**
Dipl.-Inf. Michael Gernoth

Beginn der Arbeit: 01.06.2008
Abgabe der Arbeit: 12.02.2009

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, den

Zusammenfassung

Diese Arbeit gibt einen Überblick über die MIFARE-Technologie, insbesondere MIFARE-Classic und drei konkrete Anwendungsgebiete. Die Anwendungsgebiete wurden unter Sicherheitsgesichtspunkten untersucht und verschiedene potentielle Schwachpunkte und Angriffe werden skizziert und beschrieben. Abschließend wird die MIFARE-Technologie ausgehend von diesen drei Anwendungsgebieten zusammenfassend kritisiert.

Die Arbeit baut auf den im letzten Jahr und im Dezember vorletzten Jahres bekannt gewordenen Schwachstellen in MIFARE-Classic Karten auf. Die damit einhergehenden Veröffentlichungen haben diese Arbeit inspiriert.

Inhaltsverzeichnis

Kapitel 1

Überblick über die MIFARE-Technologie

MIFARE ist ein Warenzeichen von NXP Semiconductors und stellt mit weltweit über einer Milliarde Karten die am weitesten verbreitete Produktfamilie von Kontakt-losen Smartcards dar [1]. Die Karten kommen in den verschiedensten Anwendungsgebieten zum Einsatz, so zum Beispiel als Eintrittskarten für Sportgrossveranstaltungen, in Zugangssystemen für Gebäudekomplexe, als (wegwerfbare) Fahrkarten in öffentlichen Verkehrssystemen oder als Geldkarten, zum Beispiel in den Mensen vieler deutscher Universitäten.

Die MIFARE Technologie basiert auf dem "ISO/IEC 14443 (RFID) Type A 13.56 MHz contactless smart card" Standard. Die Karten verfügen über keine eigene Stromversorgung, sondern induzieren Strom aus einem vom Lesegerät zur Verfügung gestellten Feld. Dies limitiert die Rechenleistung der Karte, macht sie jedoch langlebig, kostengünstig und zuverlässig. Alle Karten der MIFARE Familie implementieren zumindest ISO 14443-3 Typ A als Kommunikationsprotokoll.

1.1 MIFARE Classic (Standard)

Die MIFARE Classic Karte[2] ist eine Speicherkarte mit einfachen Sicherheitsmechanismen für die Zugriffskontrolle. Die MIFARE Classic 1K verfügt über 1024 Byte Speicher unterteilt in 16 Sektoren (Abbildung 1.1) zu je 4 Blöcken à 16 Byte (Abbildung 1.2). Die Sektoren können durch je zwei geheime Schlüssel A und B geschützt werden. Der letzte Block jedes Sektors enthält die Zugriffsrechte (Abbildung 1.3), Block 0 die werksseitig festgelegte, nicht veränderbare UID der Karte (Abbildung 1.4).

[1] http://www.nxp.com/products/identification/MIFARE/

[2] http://www.nxp.com/acrobat/other/identification/M001053_MF1ICS50_rev5_3.pdf

1.1.1 Aufbau der MIFARE Classic 1K

Sektor 1
Sektor 2
...
Sektor 3
Sektor 16

Abbildung 1.1: Aufbau der MIFARE Classic 1K

Block 8	Daten (16 Byte)
Block 9	Daten (16 Byte)
Block 10	Daten (16 Byte)
Block 11	Key A, Zugriffsrechte (4 Byte), Key B (je Key 6 Byte)

Abbildung 1.2: Blockaufbau der MIFARE Classic 1K

Key A	Zugriffsrechte	Key B

Abbildung 1.3: Aufbau von Block 3, 7, ... der MIFARE Classic 1K

UID (4 Byte)	Prüfsumme (1 Byte)	Herstellerdaten (11 Byte)

Abbildung 1.4: Aufbau von Block 0 der MIFARE Classic 1K

Als Verschlüsselungsalgorithmus kommt CRYPTO-1, ein selbst entwickelter proprietärer Algorithmus, zum Einsatz. Die Implementierung und Funktionsweise des Algorithmus wurde im Widerspruch zu dem Prinzip von Kerckhoff ([4]) geheimgehalten. Er wurde jedoch von Flavio D. Garcia e.a. durch genaue Protokollanalyse unter Zuhilfenahme der Reverse-Engineering-Arbeiten von Karsten Nohl und Henryk Plötz erfolgreich cryptoanalytisch untersucht und gebrochen. ([1], [2], [3], [5]).

1.1.2 Varianten der MIFARE Classic 1K

Die MIFARE Classic 4K[3] bietet 4096 Byte Speicher, unterteilt in 40 Sektoren, von denen 32 Sektoren 64 Byte groß sind und 4 Sektoren 256 Byte. Nach unten wird die Classic-Gruppe durch die MIFARE Classic Mini [4] ergänzt die 320 Byte Speicher in 5 Sektoren bietet.

[3]http://www.MIFARE.net/products/smartcardics/MIFARE_standard4k.asp
[4]http://www.MIFARE.net/products/smartcardics/MIFARE_mini.asp

1.2 MIFARE Ultralight

Die MIFARE Ultralight[5] ist die mit Abstand günstigste Karte der MIFARE Familie. Aufgrund ihrer geringen Kosten wird sie meist als Wegwerfticket eingesetzt. Sie verfügt über keinerlei kryptographischen Zugriffsschutz.

Die Karte hat insgesamt 64 Byte Speicher, organisiert in 16 Pages à 4 Byte, 54 Byte davon sind für den Benutzer verfügbar. Die verbleibenden 10 Byte werden folgendermaßen ab Werk verwendet: 7 Byte UID und 2 Byte Prüfsumme, das letzte Byte ist für karteninterne Benutzung vorgesehen. Die 54 für den Anwender nutzbaren Byte setzen sich aus 16 Lockbits, 48 Byte allgemeiner Datenspeicher und 32 One-Time-Programmable-Bits zusammen (Abbildung 1.5).

Die 16 Lockbits werden verwendet, um einzelne Pages als les-, oder schreibbar zu markieren. Zusätzlich sind noch Bits vorgesehen, um jeweils die Lockbits mehrerer Blöcke als nicht mehr veränderbar zu markieren. Diese Bits finden sich im low nibble von lock0.

Wenn ein Lockbit einmal gesetzt wurde, ist dies nicht mehr revidierbar. Die 32 One-Time-Programmable-Bits sind alle mit 0 vorinitialisiert und können einzeln auf 1 gesetzt werden. Auch diese Operation ist irreversibel. Die Bits können zum Beispiel verwendet werden, um einen Zähler von 0 bis 31 zu implementieren. Denkbar wäre hier eine Anwendung, die auf der Karte bis zu 32 Einzelfahrten in einem Verkehrssystem erlaubt. Die Zahl der Fahrten wird durch Setzen der OTP-Bits beim Kauf festgelegt und anschließend bei jeder Fahrt ein weiteres Bit gesetzt. Sobald alle Bits gesetzt wurden, ist die Karte verbraucht. Die Datenbytes können je nach Berechtigung auf den Lockbits frei geschrieben werden. Die UID der Karte ist ab Werk fest und kann nicht geändert werden.

1.3 MIFARE ProX, SmartMX

Die MIFARE ProX und SmartMX[6] sind mikroprozessorbasierte Karten, auf denen ein Betriebssystem läuft. Sämtliche Funktionalität ist programmiert; für kryptographische Anwendungen wird üblicherweise ein Kryptocoprozessor hinzugefügt. Dieser übernimmt dann die kryptographische Arbeit (meist RC4 oder RSA). Diese beiden Kartentypen sind auch als kontaktbasierte Varianten erhältlich. Je nach verwendeter Software sind diese Karten sehr vielseitig einsetzbar und versprechen eine höhere Sicherheit als die MIFARE Classic Varianten.

[5] http://www.MIFARE.net/products/smartcardics/MIFARE_ultralight.asp
[6] http://www.MIFARE.net/products/smartcardics/smartmx.asp

Page Name	0	1	2	3	Page Number
Serien-Nr.	UID 0	UID 1	UID 2	BCC 0	0
Serien-Nr.	UID 3	UID 4	UID 5	BCC 6	1
Internal/Lock	BCC 1	Internal	Lock 0	Lock 1	2
OTP	OTP 0	OTP 1	OTP 2	OTP 3	3
Data	Data 0	Data 1	Data 2	Data 3	4
Data	Data 0	Data 1	Data 2	Data 3	5
Data	Data 0	Data 1	Data 2	Data 3	6
Data	Data 0	Data 1	Data 2	Data 3	7
Data	Data 0	Data 1	Data 2	Data 3	8
Data	Data 0	Data 1	Data 2	Data 3	9
Data	Data 0	Data 1	Data 2	Data 3	A
Data	Data 0	Data 1	Data 2	Data 3	B
Data	Data 0	Data 1	Data 2	Data 3	C
Data	Data 0	Data 1	Data 2	Data 3	D
Data	Data 0	Data 1	Data 2	Data 3	E
Data	Data 0	Data 1	Data 2	Data 3	F

Abbildung 1.5: Speicherorganisation der MIFARE Ultralight; Grau unterlegte Bereiche sind Anwenderbereiche.

1.4 MIFARE DESFire

Die MIFARE DESFire basiert im Kern auf den MIFARE ProX/SmartMX Karten, kommt jedoch bereits ab Werk mit einer Software (dem DESFire operating system). Die Karte gibt es in vier Varianten: einer mit Triple-DES[7] und 4096 Byte Speicher und drei mit AES[8] und je 2048, 4096 oder 8192 Byte Speicher. Die Karten basieren auf einem 8051 Prozessor mit Triple-DES oder AES crypto-Beschleuniger. Diese beiden Blockchiffren gelten bis heute für die meisten Anwendungen als ausreichend sicher.

1.5 MIFARE Plus

Bisher nur angekündigt sind die MIFARE Plus Karten[9] als kostengünstiger Ersatz für die MIFARE Classic Varianten. Das Datenmanagement ist mit dem der MIFARE Classic Karten identisch, als Kryptoalgorithmus kommt jedoch AES mit einer Schlüssellänge von 128 Bit zum Einsatz, was eine Modifikation der alten Lesegeräte nötig macht.

[7] http://www.MIFARE.net/products/smartcardics/MIFARE_desfire.asp
[8] http://www.MIFARE.net/products/smartcardics/MIFARE_desfire_ev1.asp
[9] http://www.MIFARE.net/products/smartcardics/MIFARE_plus.asp

Für diese Arbeit sind ausschließlich MIFARE Classic und MIFARE Ultralight von Relevanz.

Kapitel 2

Verwendete Hardware

Für diese Arbeit wurden drei spezielle Geräte verwendet: ein OpenPCD, ein OpenPICC und der Proxmark III. Für alle drei Geräte sind Baupläne und Firmware unter freien Lizenzen im Netz verfügbar.

2.1 OpenPCD

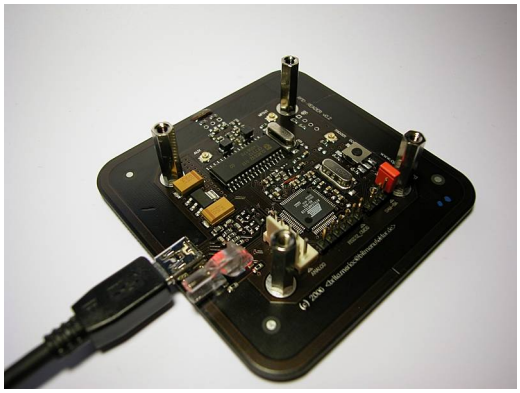

Abbildung 2.1: OpenPCD ©http://openpcd.org

Bei dem OpenPCD[1] handelt es sich um ein Gerät, das entworfen wurde, um als Empfangsgerät (sog. proximity coupling device, kurz PCD) für standardkonforme Proximity Integrated Circuit Cards (PICCs) zu fungieren. Das Ziel des Entwurfs ist es, dem Anwender vollständige Kontrolle über das RFID-Signal zu geben und verschiedene Ausgabesignale zur Verfolgung der Kommunikation anzubieten. Die Hardware wurde von Harald Welte, Milosch Meriac und Brita Meriac entworfen.

Das Design basiert auf dem RC632 Multiple Protocol Contactless Reader IC der Firma Philips, welcher die ISO 14443 A und B, ISO 15693, MIFARE und

[1] http://www.openpcd.org

ICODE Protokolle implementiert. Zusätzlich ist auf dem Gerät ein 23-bit-ARM-Mikroprozessor (AT91SAM7S128) verbaut, um den Philips-Chip anzusteuern und zu kontrollieren. Dieser verfügt über eine RISC CPU, 32 Kilobyte SDRAM und 128 Kilobyte Flashspeicher. Ansprechbar ist das Gerät über eine Mini-USB-Buchse und einen 20-Pin JTAG Anschluss.

2.2 OpenPICC

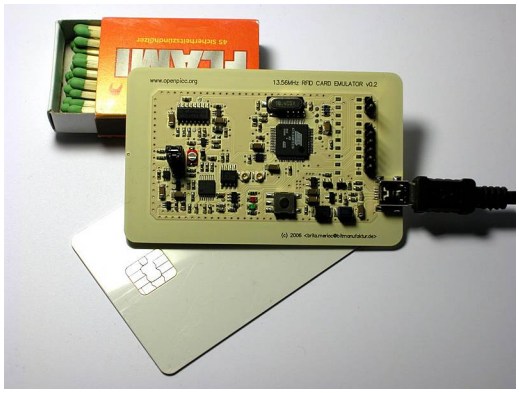

Abbildung 2.2: OpenPICC v0.2 (ohne Modifikationen) ©http://openpicc.org

Der OpenPICC[2] ist das Schwesterprojekt und Gegenstück des OpenPCD. Es emuliert 13.56 MHz basierte RFID Transponder oder Smartcards (sog. Proximity Integrated Circuit Cards (PICCs)). Unter anderem kann der OpenPICC dazu verwendet werden, ISO 14443-kompatible Karten wie die MIFARE-Karten zu simulieren.

Auf dem OpenPICC ist ebenfalls eine 32-bit-ARM-CPU der AT91SAM7-Familie verbaut (der AT91SAM7S256), der über 256 Kilobyte Flash-Speicher sowie 64 Kilobyte SDRAM verfügt.

Das ursprüngliche Design des OpenPICC hatte Timing-Probleme, die durch einige von Henryk Plötz vorgeschlagene Änderungen am Design gelöst wurden. Die Anleitung zur Modifikation der Hardware findet sich im Netz und wurde auch an dem von mir verwendeten OpenPICC vorgenommen.

Es existiert Firmware, die es ermöglicht, die Kommunikation vom Lesegerät zu einer MIFARE-Karte mit dem OpenPICC mitzulesen. Zusätzlich wurde die Firmware von mir modifiziert, um eine beliebige MIFARE-UID zu emulieren.

Mit dem OpenPICC ist es mir gelungen, die Kommunikation unidirektional vom Leser zur Karte mit zuhören. Die Gegenrichtung von der Karte zum Leser hat jeweils schon nach einigen Bits so starken Drift auf dem Signal, dass kein Mithören mehr möglich war. Durch eine speziell dafür ausgelegte Antenne ist es eventuell

[2]http://www.openpicc.org

möglich, das Problem zu lösen. Mir sind Fälle bekannt, in denen es gelungen ist, bidirektionale Kommunikation nur mit dem OpenPICC abzuhören, dies konnte ich jedoch nicht reproduzieren. In Abbildung 2.3 lässt sich das Problem erkennen. Man sieht hier zuerst das Signal vom Leser an die Karte und anschließend die Antwort der Karte.

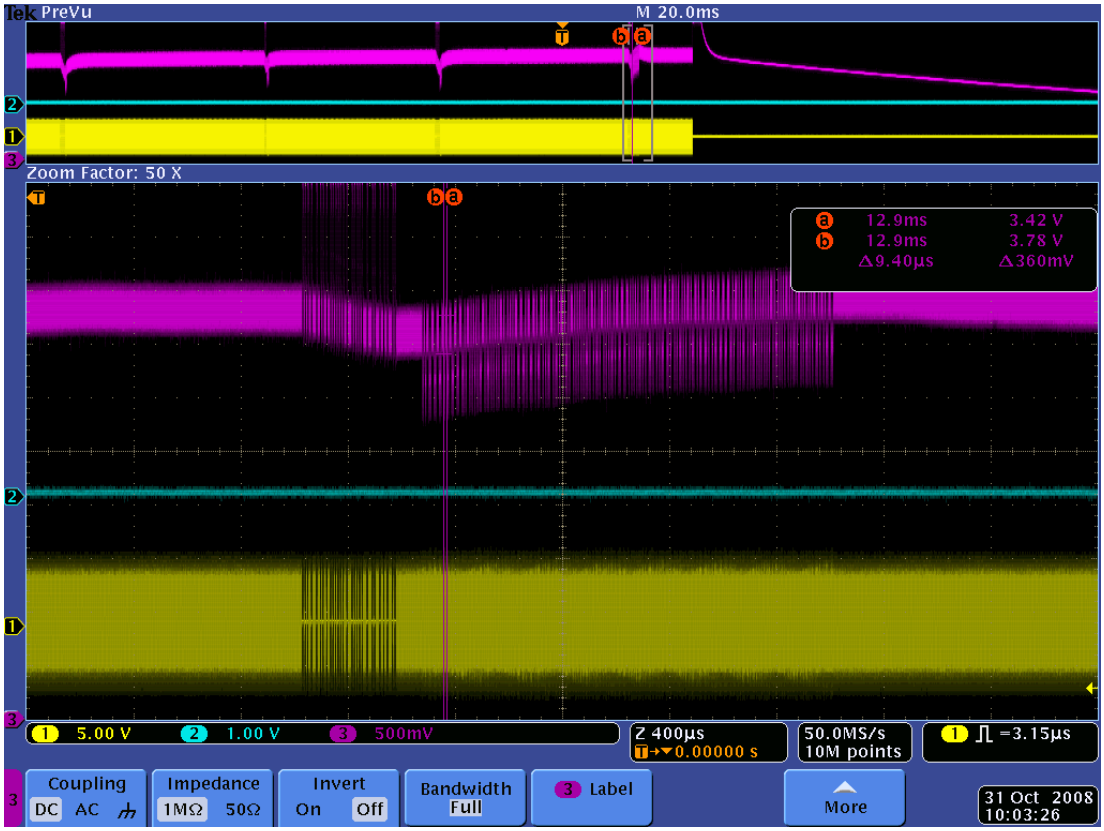

Abbildung 2.3: Drift-Probleme am OpenPICC

2.3 Proxmark III

Der Proxmark 3[3] ist die dritte Revision eines von Jonathan Westhues entwickelten Testgerätes für RFID-Karten im hoch- (etwa 13.56 MHz) und niederfrequenten (etwa 125 kHz) Bereich. Das Gerät kann einen Leser oder ein Tag emulieren, bidirektional die Kommunikation zwischen einem Lesegerät und einem Tag mithören und die empfangenen Signale weiterverarbeiten.

Auf dem Proxmark ist wie auf dem OpenPICC ein AT91SAM7S256 Mikrocontroller verbaut. Des Weiteren ein FPGA von Xilinx (Spartan II XC2S30). Eintreffende Signale werden im A/D-Wandler digitalisiert und anschließend im

[3] http://www.proxmark.org

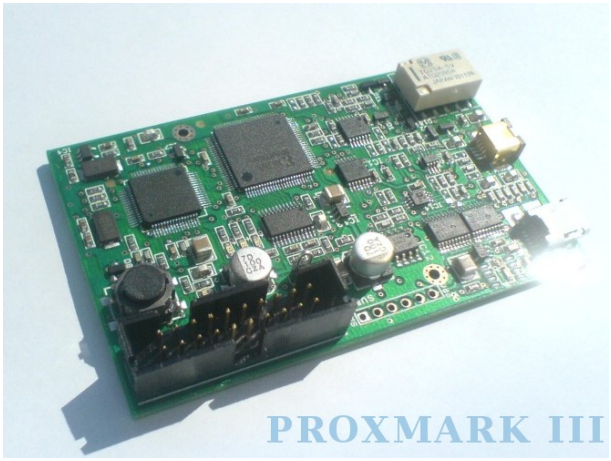

Abbildung 2.4: Proxmark 3 ©http://proxmark.org

FPGA verarbeitet. Die Verwendung des FPGAs als digitaler Signalprozessor entlastet den general-purpose Mikrocontroller vor allem bei hochfrequenten Signalen.

Der Proxmark hat im Gegensatz zu den beiden anderen Geräten keine interne Antenne. Für die MIFARE Karten hat sich eine einfache Antenne als vollkommen ausreichend erwiesen (teilweise sogar als zu gut, bei manchen Lesegeräten ließ sich die Antenne als Störsender verwenden). Vier rechteckige Windungen aus Spulendraht mit einer Kantenlänge von 65mm und 13mm haben zu guten Ergebnisse geführt. Eine so kleine Antenne lässt sich auch bequem auf eine MIFARE Karte aufkleben, falls die zu untersuchenden Geräte die Karte einziehen und das Gehäuse die Signale zu stark abschirmt. Es ist bei der Montage der Antennen darauf zu achten, dass die Anschlussdrähte möglichst kurz sind, aber die Antenne nicht zu nahe am Proxmark betrieben wird, da die Signalleitungen auf der Platine sonst den Empfang stören. Ein Abstandhalter aus etwa 25mm starken Schaumstoff hat diese Probleme gelöst (vgl. Abbildung 5.2).

Kapitel 3

Moskauer Metrofahrkarten

Die Moskauer Metro transportiert pro Tag knapp 10 Millionen Fahrgäste. Nach einer kurzen Testphase werden dort seit Sommer 2007 RFID Karten als Wegwerffahrkarten eingesetzt. Von einem Aufenthalt in Moskau habe ich zwei dieser Karten für eine Analyse mitgebracht (Abbildung 3.1). Jede Karte ist auf der Rückseite mit einer zehn-stelligen, dezimalen sog. “Bilet Nummer” versehen, die jedoch von der UID der Karte verschieden ist (Abbildung 3.2).

Abbildung 3.1: Fahrkarte der Moskauer Metro

3.1 Analyse der Fahrkarten

Bei den Karten handelt es sich um MIFARE Ultralight Karten, wie leicht an der sieben Byte langen UID zu erkennen war. Die vorliegenden Karten waren zu

Abbildung 3.2: Rückseite der Karte mit “Bilet Nummer”

Beginn der Analyse wie folgt belegt:

Bezeichner	**urspr. Fahrtzahl**	**verbl. Fahrtzahl**
K1	1	0
K2	1	1

Im einem ersten Schritt fiel bei einem Vergleich der Karten K1 und K2 (Abbildung 3.3) auf, dass die OTP-Bits auf beiden gleich sind. Die Anzahl der verbleibenden Fahrten wird also nicht durch die OTP-Bits codiert. Weiterhin ist zu erkennen, dass alle Blöcke im Moment als rein lesbar markiert sind, aber nicht fest gesperrt. Es sind also alle Datenblöcke der Karte noch modifizierbar.

Bei genauerer Betrachtung der Datenblöcke stellt man fest, dass K1 und K2 jeweils in Page 9 und A sowie in Page D und E identische Information tragen, was vermutlich der Datensicherheit durch Redundanz dienen soll. Das letzte Byte in Page 4 und die ersten 5 nibbles in Page 5 stellen die “Bilet Nummer” dar.

Die Bedeutung der restlichen Daten sowie der Daten in Page 9, A, D, und E ist noch unklar. Vermutlich enthalten sie in verschlüsselter oder unverschlüsselter Form neben der Anzahl der verbleibenden Fahrten noch andere Information. Vorstellbar wären zum Beispiel Zeitstempel für Ausgabe der Karte, letzte Benutzung oder Verkaufsstelle der Karte und ursprüngliche Fahrtenanzahl.

Die Unterschiede zwischen den Karten in Page 0 sind durch die unterschiedliche UID erklärbar.

Byte 0	Byte 1	Byte 2	Byte 3	Page
04	51	b9 (b8)	64 (65)	0
89	3c	25	80	1
10	48	f0	ff	2
ff	ff	ff	fc	3
41	87	80	ea	4
49	5a	c8 (d8)	17	5
dc	00	00	00	6
00	00	00	00	7
17	d8	05	00	8
00	00	01 (04)	ab (d8)	9
f3 (03)	ac (81)	5b (0b)	2a (e2)	A
00	00	00	00	B
17	d8	05	00	C
00	00	01 (04)	ad (d8)	D
f3 (03)	ac (81)	5b (0b)	2a (e2)	E
00	00	00	00	F

Abbildung 3.3: Vergleich von K1 und K2 (in Klammern abweichende Werte bei K2)

3.2 Replayangriff

Da die Information über die verbleibende Fahrtzahl nicht in den OTP-Feldern gespeichert sein kann und alle anderen Datenfelder der Karte mit einem entsprechenden Lesegerät frei gelesen und geschrieben werden können ist ein Replayangriff möglich. Man liest den Inhalt der Datenblöcke einer frisch erworbenen Karte aus und sichert ihn. Jetzt verbraucht man die Fahrten auf der Karte und spielt den gesicherten Inhalt anschließend wieder zurück. Die Karte enthält jetzt wieder die gleiche Information wie zum Zeitpunkt der Datensicherung und ist von einer neu erworbenen Karte nicht zu unterscheiden. Dies ermöglicht es, bei einmaligem Erstehen einer Karte beliebig oft ohne zusätzliche Kosten zu fahren.

3.3 Gegenmaßnahmen

Eine denkbare Gegenmaßnahme ist die Verwendung eines Backend-Systems, das solche Manipulationen durch Führung eines Schattenkontos entdeckt und einen Alarm auslöst. Falls kein solches System verwendet wird, sondern die Information über die verbleibende Fahrtzahl ausschließlich auf der Karte verwaltet wird ist die Manipulation nicht nachweisbar. Ein abschließender Test, ob eine Manipulation entdeckt wird, ob also ein solches Backend-system in Moskau verwendet wird, konnte noch nicht durchgeführt werden.

Kapitel 4

Zugangssystem der FAU (Siport)

Innerhalb der FAU kommt ein Türschliesssystem zum Einsatz, das auf MIFARE basiert. Das System ermöglicht zum Beispiel Studenten an der technischen Fakultät den Zugang und die Nutzung des CIP-Pools der Informatik auch außerhalb der üblichen Öffnungszeiten, wenn sie ein MIFARE-Token einmalig dafür haben registrieren lassen. Üblicherweise werden hier die Mensakarten (vgl. 5.1) oder die Kopierkarten der Universität verwendet, die beide MIFARE-Karten sind und üblicherweise von jedem Studenten besessen werden. Mitarbeiter, beispielsweise des RRZE, bekommen teilweise spezielle Schlüsselanhänger ausgehändigt, die einen MIFARE-Chip integriert haben und ebenfalls am Türschliesssystem registriert werden können (Abbildung 4.1). An den ins Türschliesssystem integrierten

Abbildung 4.1: Schlüsselanhänger mit MIFARE-Chip

Türen befinden sich Lesegeräte, die auf ein in die Nähe gebrachtes MIFARE-Token reagieren und eine Anfrage an ein Backend-system leiten, das entscheidet ob der Besitzer dieses Tokens durch die Türe gehen darf oder nicht. Je nachdem leuchtet eine rote oder grüne LED am Leser auf und die Tür wird entriegelt oder bleibt verschlossen. Das System (Hardware und Software) wird von Siemens unter dem Namen "SIPORT"[1] vertrieben.

[1] www.buildingtechnologies.siemens.de/NR/exeres/41847AD3-757A-4522-AEA0-BD1689AA46BE.htm

4.1 Identitätsdiebstahl

Das System verwendet zum Lesen der Karten keine der Sicherheitsfeatures der MIFARE-Karten. Es wird einfach die UID der Karte ausgelesen und auf Grundlage dieser UID die Entscheidung getroffen, wer Zutritt erhält und wer nicht. Weitere Daten sind auf der Karte nicht abgelegt, die kryptographischen Fähigkeiten werden nicht genutzt.

Dies macht es extrem einfach für jemanden, der über einen Emulator verfügt, unter der Identität einer anderen Person durch Türen zu gehen oder Bereiche zu betreten, zu denen er eigentlich keinen Zutritt hat. Der Emulator wird mit einer entsprechenden UID programmiert, die er sendet, sobald er das Feld eines Lesers erkennt. Wenn die Person, die eigentlich im Besitz der Karte mit dieser (vermeintlich einzigartigen) UID ist, berechtigt ist, die Tür zu öffnen, wird sich die Tür auch für den Besitzer des Emulators öffnen.

Es wäre möglich, UIDs zu raten, da innerhalb eines Systems mit einer gewissen Wahrscheinlichkeit nah beieinander liegende UIDs vergeben werden und man so aus einer einzigen bekannten UID andere ableiten könnte. Einfacher ist es aber, sich die UID, über die man verfügen möchte, gezielt zu beschaffen. Das Token des legitimen Besitzers sendet die UID aus, sobald es sich im Bereich eines beliebigen MIFARE-Lesers wähnt. Das kann von einem Angreifer ausgenutzt werden um sich mit einem kleinen Leser in die Nähe des Opfers zu begeben und so dessen, dann ausgesendete, UID zu erlangen. Möglichkeiten dafür gibt es nahezu unendlich viele: im Bus, auf der Toilette oder bei einer Begegnung im Korridor.

Das System kann im Hintergrund nicht unterscheiden, ob es mit einem Emulator redet oder mit einem gültigen Token. Sämtliche Log-Dateien oder Ähnliches werden nach einem Sicherheitsvorfall auf den legitimen Besitzer des Tokens verweisen und diesen unverdient zum Verdächtigen machen.

Im folgenden findet sich ein kurzer Auszug, in dem gezeigt wird, wie einem OpenPICC eine zuvor ausgelesene UID (0xd240a3ad) einprogrammiert wird. Zusätzlich lässt sich ein Überblick über die sonstigen Funktionen des OpenPICC gewinnen:

```
Waiting for carrier. Carrier detected.
?Command received:?
Got command 63 with args
 ****************************************************
 * OpenPICC USB terminal                            *
 * (C) 2007 Milosch Meriac <meriac@openbeacon.de>   *
 * (C) 2007 Henryk Ploetz <henryk@ploetzli.ch>      *
 ****************************************************
 * Version
 * compiled 20090112-180216 by spjsschl@faui03a
 * running on OpenPICC v0.4 (Karsten's edition)
```

```
 *
 * thru - test throughput
 * c    - print configuration
 * +,-  - decrease/increase comparator threshold
 * #    - switch clock
 * l    - cycle LEDs
 * p    - print PIO pins
 * r    - start/stop receiving
 * z 0/1- enable or disable tc_cdiv_sync
 * i    - inhibit/uninhibit PLL
 * !    - reset tc_cdiv_sync
 * q    - start rx
 * f    - start/stop field meter
 * d div- set tc_cdiv divider value 16, 32, 64, ...
 * j,k  - increase, decrease fdt_offset
 * a    - change load modulation level
 * g 0/1- disable or enable SSC_DATA through gate
 * 9    - reset CPU
 * s id - set new MIFARE classic id (supply as 8-char-hex-string)
 * ?,h  - display this help screen
 *
 ******************************************************
Command received:s d240a3ad
Got command 83 with args d240a3ad
Setting new ID d240a3ad ...
done.
```

Dieser Vorgang lässt sich noch weiter automatisieren. Zum Beispiel ist es mit dem Proxmark3 möglich, eine UID erst auszulesen und diese ohne weitere Interaktion anschließend an irgendwelchen Lesegeräten wieder abzuspielen (Replay-Angriff). In Abbildung 4.2 ist zu sehen, wie eine Tür mit dem OpenPICC, der mit einem Batteriesatz versehen ist, geöffnet wird.

4.2 Schwachstellen in der Umsetzung

4.2.1 PINS

Einige Türen sind zusätzlich durch PIN-Pads gesichert, an denen als zusätzliche Schranke eine gültige PIN eingegeben werden muss, bevor die Tür sich öffnet (Abbildung 4.3).

In der Datenbank des Zutrittskontrollsystems sind die PINs der Benutzer im Klartext abgelegt. Lesender Zugriff auf die Datenbank ist daher ausreichend, um sich die PIN (und auch die Token-UID) eines Zutrittsberechtigten zu verschaffen.

Abbildung 4.2: OpenPICC mit Batterien beim Öffnen einer Tür mit einer zuvor einprogrammierten UID

Eine weitere Schwachstelle ist die Default-PIN jedes Benutzers (000000). Diese wird als gültige PIN an den Kartenlesern akzeptiert, sollte aber durch das System abgewiesen werden, um eine nicht-triviale PIN zu erzwingen.

4.2.2 Passwörter

Die Benutzernamen und Passwörter zum Zugriff auf die Datenbank werden durch Anwenden der Funktion XOR in der Datenbank abgelegt. XOR ist keine Verschlüsselung, sondern eine Operation, welche durch eine weitere Anwendung rückgängig gemacht werden kann. Die XOR-Operation ist eine Involution, also selbstinvers (Es gilt: $a \oplus b = c$ und anschließend $c \oplus b = a$ und $c \oplus a = b$).

Da ein Operand der Funktion allgemein bekannt ist, nämlich der Login-Name des Benutzers, kann der zweite Operand, das Passwort des Benutzers, aus der Datenbank rekonstruiert werden. Um an die benötigten Daten für die Rekonstruktion der Passwörter zu gelangen, ist ein lesender Zugriff auf die Datenbank ausreichend. Statt der XOR-Funktion sollte eine gute Hash-Funktion (beispiels-

weise aus der SHA-Familie) für das Passwort genutzt werden, da diese nicht einfach umkehrbar ist und das Passwort aus dem Datenbankinhalt nicht wiederhergestellt werden kann.

4.2.3 Ungeschützte Leserkabel

Die Kabel an der RRZE Bunkertüre und an anderen Orten verlaufen ungesichert in einem Aufputzkabelkanal welcher in Sekunden ohne Hilfsmittel geöffnet werden kann. Die Kommunikation findet über ein einfaches, unverschlüsseltes, serielles Bus-Protokoll statt, welches mit geringem Hardwareaufwand mitgehört werden kann. Auch ein nichtinvasives Abhören durch das induzierte Feld des Kabels scheint möglich. Die benötigte Hardware lässt sich unauffällig in den vorhandenen Kabelkanälen unterbringen, da diese genug Platz bieten (Abbildung 4.3). Über diesen Bus wird neben der Token-ID des Benutzers auch die PIN unverschlüsselt übertragen und kann so auch Personen zugänglich werden, die nicht über einen Datenbank-Login verfügen.

4.2.4 unverschlüsselte Kommunikation zwischen Leser und Karte

Wie oben bereits erwähnt, nimmt der Leser die Token-ID der Karte über eine unverschlüsselte Kommunikationsverbindung entgegen, obwohl die ID auch über eine verschlüsselte Verbindung mit den allgemein bekannten Herstellerschlüsseln A0A1A2A3A4A5 bzw. FFFFFFFFFFFF abgefragt werden kann.

Dies ist ein grundlegender Design-Fehler, der einen Angriff auch ohne die in letzter Zeit bekannt gewordenen Angriffsmöglichkeiten auf MIFARE-classic Karten ermöglicht. Die UIDs werden üblicherweise nicht geheimgehalten, sie stehen im Klartext in der Datenbank, was es für alle Personen mit lesendem Zugriff auf die Datenbank noch einfacher macht, einen Emulator mit einer gültigen UID zu versehen. Ein Abhören der UID ist nicht mehr nötig.

4.2.5 Ungeschützte Netzwerkkabel

So genannte MBOXen stellen die Verbindung zwischen Lesern und dem dahinter hängenden Netzwerk mit Datenbankserver dar. Eine der MBOX nachgeschaltete VPN-Box sorgt dafür, dass die Kommunikation zum Datenbankserver verschlüsselt abläuft (Abbildung 4.4). Die Kommunikation über das Kabel zwischen Leser und MBOX sowie zwischen MBOX und VPN-Box ist jedoch unverschlüsselt.

Einige Netzwerkkabel zwischen MBOX und VPN-Box sind von außen erreichbar, weshalb sich ein Angreifer in das unverschlüsselte Netz zwischen MBOX und VPN-Box einhängen kann. Auf diese Weise hat er Kontrolle über alle Türen der

Abbildung 4.3: Leser mit integriertem PIN-Pad und geöffnetem Kabelkanal

betreffenden MBOX und kann auch das Zutrittskontrollsystem auf dem Server angreifen.

Abbildung 4.4: MBOX (unten) und in Switchbox integrierte VPN-Box (oben)

Kapitel 5

Mensakarten der FAU (Girovend)

Abbildung 5.1: Weiße Mitarbeitermensakarte

5.1 Funktionsweise

Das Girovend-system wird in der FAU hauptsächlich eingesetzt, um damit in der Mensa und an Automaten bargeldlos zu zahlen. Üblicherweise besorgen sich Studenten zu Beginn ihres Studiums eine gelbe, so genannte Mensakarte gegen ein geringes Geldpfand. Auch für an der Universität Beschäftigte stehen Karten zur Verfügung. Diese sind jedoch weiß (Abbildung 5.1) statt gelb, vermutlich um optisch erkennbar zu machen dass für Besitzer dieser Karten andere Preise

gelten. Es stehen frei zugängliche Aufladeautomaten zur Verfügung, an denen das Guthaben mit Geldscheinen (5, 10 und 20€) bis zu einem Maximum von 50€ aufgeladen werden kann. An der "Kasse" wird dann der Preis des Essens abgebucht. An den Aufladeautomaten kann auch einfach nur der noch auf der Karte stehende Betrag eingesehen werden.

5.2 Angriff auf die Schlüssel

Bei den Mensakarten handelt es sich um MIFARE Classic 1K. Die Vermutung, der Geldbetrag sei auf den Karten gespeichert, hat die Tatsache nahe gelegt, dass zum Beispiel bei den Getränkeautomaten in der WISO eine Außenanbindung an ein Backend-system, das die Buchhaltung erledigt, definitiv nicht vorhanden ist. Das macht die Information auf den Karten zu einem Angriffsziel. Um die Karten manipulieren zu können, benötigt man die Schlüssel, die zum Lesen und Schreiben notwendig sind.

Ein erster Versuch hat schon vor Jahren gezeigt, dass keiner der sog. Standardschlüssel verwendet wird, mit dem die Karten ausgeliefert werden. Ein Bruteforceangriff erscheint aussichtslos, da ein Authentifizierungsversuch mit Standard-Hardware etwa 25ms ([6]) dauert. Da jeder der beiden möglichen Schlüssel eine Länge von 48 bit hat ergibt sich als maximal nötige Zeit bis der Schlüssel gefunden wird etwa 223.000 Jahre ($2^{48}\text{keys}\frac{25\text{ms}}{\text{key}} \approx 223.000$ years). Da alle Schlüssel mit derselben Wahrscheinlichkeit gewählt wurden, darf man erwarten, schon nach etwa 111.000 Jahren einen Schlüssel gefunden zu haben ($\frac{223.000\text{ years}}{2} \approx 111.000$ years).

Flavio D. Garcia e.a. haben eine einfache Methode gezeigt, wie man aus der Kommunikation zwischen Lesegerät und Mensakarte den Schlüssel des Crypto-1 Cipher berechnen kann ([2]). Die Berechnung des Schlüssels habe ich unter Zuhilfename von Henryk Plötzs Diplomarbeit, der dort eine Implementierung ([1]) des Crypto-1 veröffentlicht hatte, und eines Hackers, der unter dem Pseudonym "Bla" eine fast vollständige Implementierung des Angriffs ins Netz gestellt hatte, hatte[1] implementiert.

5.3 Abhören der Kommunikation und Schlüsselberechnung

Mit einem Proxmark3 wurde solange versucht, die Kommunikation zwischen Aufladeautomaten und Mensakarte (beim Auslesen des Betrages) mitzuhören, bis genug fehlerfreie Information vorhanden war, um jeweils den Schlüssel A für die beiden Sektoren 3 und 4 zu berechnen (Abbildung 5.2).

Auszug aus der mitgeschnittenen Kommunikation (aus diesem Auszug ist lediglich der bekannte Standardschlüssel für Sektor 1 berechenbar (A0A1A2A3A4A5)):

[1] http://crapto1.googlecode.com/files/crapto1-v0.6.tgz

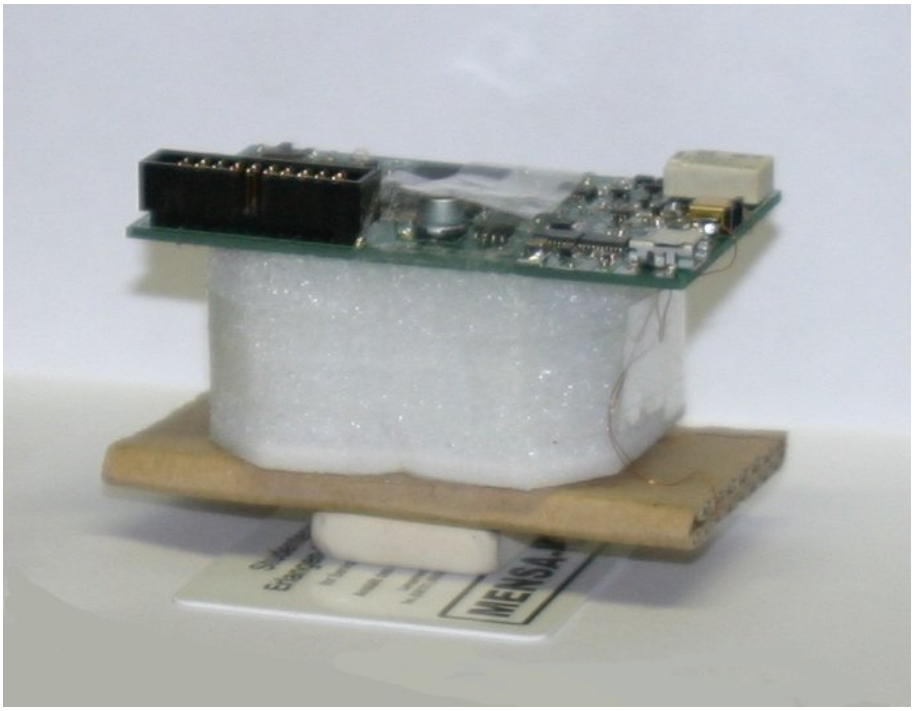

Abbildung 5.2: Proxmark 3 mit Antenne und einem Radiergummi, um den Abstand zur Mensakarte zu halten

```
+      0:   0: TAG 04  00
+    504:    :     93  20
+     64:   0: TAG 44  14  74  0b  2f
+   1216:    :     93  70  44  14  74  0b  2f  07  5a
+     64:   0: TAG 08  b6  dd
+  50204:    :     60  01  7c  6a
+    112:   0: TAG 40  d4  7a  ed
+    952:    :     1a! 1d! fd  a0  0d! 95! f0! ad
+     64:   0: TAG 09! e6! 9e! 62
+    680:    :     fc  1b  04! ab
+     72:   0: TAG 2a  c6  77  1e  fd! e3  f3! b4  c2! 7b! ee
   fe! b0  b7  ce  b9  8f! 17!
+   1840:    :     3d  f1  0e  3f
+     72:   0: TAG 8a  e7! 6f  d0  d9! 1e  da  53  ec  a3  ca!
   5d  23! 54! 50! a1  92! 8a
+   1848:    :     ec! 9b  42  d4
+     72:   0: TAG fa  34  57! 3e! 2c  8f  0f! 10! 72! 26  42
   78! 6b  23  7b  d8! bb! e6!
+  79073:    :     bc  7b  89  0b
```

!-Zeichen markieren Bytes, die mit CRC-Fehler empfangen wurden. CRC-Fehler sind in den Teilen, die verschlüsselt übermittelt werden, relativ häufig, da die CRC-Operation vor der Verschlüsselungsoperation angewendet wird. Abbildung 5.4 gibt eine genauere Aufschlüsselung der Kommunikation, Abbildung 5.3 eine Übersicht über das Kommunikationsprotokoll; die in 5.4 nicht aufgeschlüsselten Daten sind bereits Daten des gelesenen Sektors.

1. Leser wiederholt "Request card"
2. Karte bekommt Strom und meldet sich
3. Leser fragt ab, welche Karten im Feld sind
4. Karten antworten mit UID
5. Leser selektiert eine Karte über UID
6. Karte antwortet mit Typ (MIFARE Classic)
7. Leser schickt Authentifizierungsanfrage
8. Karte antwortet mit Challenge
9. Leser antwortet mit Challenge und Response
10. Karte schickt Response an Leser

Abbildung 5.3: Übersicht über das Kommunikationsprotokoll

Step	**Sender**	**base$_{16}$**	**Bedeutung**
01	Leser	26	req type A (oben nicht zu sehen)
02	Karte	04 00	Antwort auf req type A
03	Leser	93 20	select
04	Karte	44 14 74 0b 2f	UID (4 Byte), Checksumme (1 Byte)
05	Leser	93 70 44 14 74 0b 2f 07 5a	select(UID)
06	Karte	08 b6 dd	MIFARE 1K
07	Leser	60 01 7c 6a	authentifiziere(block 01)
08	Karte	40 d4 7a ed	challenge Karte
09	Leser	1a! 1d! fd a0 0d! 95! f0! ad	Challenge u. Antwort Leser (je 4 Byte)
10	Karte	09! e6! 9e! 62	Antwort Karte

Abbildung 5.4: Kommunikationsprotokoll zwischen Karte und Leser am Beispiel (grau unterlegte Bereiche sind verschlüsselt)

Berechnung der Schlüssel:

```
[21:19:13][spjsschl@asso:~/crapto1]$ ./mensatool -c sniffedmensa1
Key: XXXXXXXXXXXX
[21:19:15][spjsschl@asso:~/crapto1]$ ./mensatool -c sniffedmensa2
Key: XXXXXXXXXXXX
```

(Die Schlüssel sind hier unkenntlich gemacht)

Bei genauerer Analyse der Karte fiel auf, dass die interessanten Sektoren die Sektoren 3 und 4 sind und die Zugriffsrechte jeweils so gesetzt sind, das die Kenntnis von Schlüssel B auch für den schreibenden Zugriff nicht nötig ist. Weiterhin wurde durch eine kleine Testreihe klar, dass sich der Inhalt des Sektor 3 bei Änderungen des Betrages nicht ändert. In Sektor 4 verändern sich die Blöcke 0x10 und 0x11.

Mit einem kleinen Tool kann man sich die Sektoren 3 und 4 genauer ansehen. Es ist zu sehen, dass in Sektor 3 anscheinend 24 Bytes Daten verwendet werden, in Sektor 4 40 Bytes.

```
[21:01:19][spjsschl@asso:~/crapto1]$ ./mensatool -d
opening layer2 handle
running layer2 anticol(_open)
running layer3 (ats)
we now have layer3 up and running
Authenticating sector 3
MIFARE auth succeeded!
Reading sector 3
        Reading block 0xc
        Reading block 0xd
        Reading block 0xe
        Reading block 0xf
Authenticating sector 4
MIFARE auth succeeded!
Reading sector 4
        Reading block 0x10
        Reading block 0x11
        Reading block 0x12
        Reading block 0x13


                -- Sector 0x3 --
 f7 ee b7 1d 3f 2e d2 83  bd 47 d5 45 51 e7 fd dd
 af cf 90 67 da 1f 40 7d  00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
 00 00 00 00 00 00 7f 07  88 69 00 00 00 00 00 00
```

```
                -- Sector 0x4 --
47 71 1d ee 23 ae 7b 17  14 a0 4e 8e e0 32 6b 47
71 df e3 0e a0 4b 58 e4  26 0b 79 be 29 72 37 c7
c0 41 ed 85 f5 fd c5 3a  00 00 00 00 00 00 00 00
00 00 00 00 00 00 7f 07  88 69 00 00 00 00 00 00
```

Der Geldbetrag ist nicht im Klartext (zum Beispiel binär) auf der Karte abgelegt, sondern in einem besonderen Datenformat. Die Tatsache, dass sich immer große Datenmengen auf einmal ändern, wenn eine Transaktion auf der Karte vorgenommen wird, liess auf die Verwendung eines weiteren Ciphers schliessen.

5.4 Datenformat

Solange das Datenformat und der Cipher nicht bekannt sind, lässt sich die Karte nicht gezielt manipulieren. Glücklicherweise konnte ich mir eine lange Reihe von ermüdenden Tests um das Datenformat herauszufinden sparen, da Klaus Stengel via Google die Software der Girovendaufladeautomaten gefunden hat. Er hat die Software (eine .NET-Anwendung) disassembliert und anschliessend in C reimplentiert. Wie auch bei Java liegt bei .NET der Anwendungscode in einer Zwischensprache vor, der Common Intermediate Language (CIL), der in einer virtuellen Maschine zur Laufzeit in echte Maschinenanweisungen übersetzt wird. Mittels eines Disassemblers kann man aus der CIL, die noch alle Symbolnamen enthält, eine menschenlesbare Form erzeugen (in der sogenannten ILAsm Syntax). Dass nun alle Symbolnamen sowie die Opcodes für eine virtuelle Stackmaschine in lesbarer Form vorlagen, hat den reverse-engineering-prozess extrem vereinfacht.

Die .NET-Dokumentation und einschlägige Seiten sind sich hier oft uneins, ob CIL nur ein Binärformat oder auch Instruktionen beschreibt, und CIL und ILAsm werden oft austauschbar gebraucht. Ich habe mich nach der (leider auch nicht doppeldeutungsfreien) Verwendung der Begriffe im “ECMA 335 - Standard” gerichtet[2].

Nachdem der recht schwache Cipher bekannt und reimplementiert war, war es möglich, die Karteninhalte in korrekter Form auszulesen und das aus der .NET-Anwendung extrahierte Datenformat zu verifizieren.

```
[23:33:23][spjsschl@asso:~/crapto1]$ ./mensatool -d
opening layer2 handle
running layer2 anticol(_open)
running layer3 (ats)
```

[2]http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-335.pdf

```
we now have layer3 up and running
Authenticating sector 3
MIFARE auth succeeded!
Reading sector 3
        Reading block 0xc
        Reading block 0xd
        Reading block 0xe
        Reading block 0xf
Authenticating sector 4
MIFARE auth succeeded!
Reading sector 4
        Reading block 0x10
        Reading block 0x11
        Reading block 0x12
        Reading block 0x13


                -- Sector 0x3 --
 03 58 00 01 00 7e 60 01  00 00 00 00 00 00 00 00
 00 01 01 00 00 00 14 23  00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
 00 00 00 00 00 00 7f 07  88 69 00 00 00 00 00 00


                -- Sector 0x4 --
 39 00 00 4d 03 1a 71 61  39 00 00 4e 03 1a 71 70
 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
 0f 00 00 00 00 00 0f 00  00 00 00 00 00 00 00 00
 00 00 00 00 00 00 7f 07  88 69 00 00 00 00 00 00
```

Das Datenformat in Sektor 3 der Karten sieht aus wie in Abbildung 5.5 dargestellt. Alle Daten sind in LSB abgelegt, es sei denn, es wird explizit auf eine andere Byte Order hingewiesen. Die "Category-ID" oder die "Tariff-ID" werden dazu verwendet, zwischen weißen Mitarbeiterkarten (jeweils 0x03) und gelben Studentenkarten zu unterscheiden (0x01). Alle Bytes der "UIC" sind 0x00 auf allen Mensakarten, was darauf schließen lässt, dass das Feld nicht verwendet wird.

Das Datenformat in Sektor 4 der Karten sieht aus wie in Abbildung 5.6 dargestellt.

Der Betrag ist als BCD (binary-coded-decimal) in Euro Cent (Währungssymbol 0x1a) abgelegt. Der Sektor 4 enthält zweimal das Tupel (Betrag, Benutzungszähler, Währung, Prüfsumme). Bei jeder Aktion, die ein Automat mit der Karte durchführt, wird das Tupel mit dem älteren Transaktionszähler aktualisiert, der Transaktionszähler um zwei inkrementiert und das Tupel neu geschrie-

Start-Byte	End-Byte	Länge	Beschreibung
0	0	1	Card Format
1	4	4	Site Code (Byte Order: 1234 → 2143)
5	7	3	Card ID (Byte Order: 123 → 321)
8	14	7	UIC
15	15	1	Null-Byte
16	16	1	Card Type
17	17	1	Tariff-ID (sic!)
18	19	2	Category ID
20	21	2	Null-Bytes
22	23	2	Prüfsumme (Bytes 0-21)

Abbildung 5.5: Datenformat in Sektor 3, (24 Bytes, Start-Byte und End-Byte jeweils inklusive

Start-Byte	End-Byte	Länge	Beschreibung
0	2	3	Betrag 1
3	4	2	Benutzungszähler 1
5	5	1	Währung
6	7	2	Prüfsumme (Bytes 0-5)
8	10	3	Betrag 2
11	12	2	Benutzungszähler 2
13	13	1	Währung
14	15	2	Prüfsumme (Bytes 8-13)
16	17	2	TOKENS1 Period 1
18	18	1	TOKENS1 1
19	19	1	LIMIT1 1
20	21	2	LIMIT2 1
22	23	2	Prüfsumme (Bytes 16-21)
24	25	2	TOKENS1 Period 2
26	26	1	TOKENS1 2
27	27	1	LIMIT1 2
28	29	2	LIMIT2 2
30	31	2	Prüfsumme (Bytes 24-29)

Abbildung 5.6: Datenformat in Sektor 4, (40 Bytes, Start-Byte und End-Byte jeweils inklusive

ben (auch wenn der Betrag nicht verändert wurde).

Ebenfalls zweimal findet sich das Tupel (TOKENS1 Period, TOKENS1, LIMIT1, LIMIT2, Prüfsumme). Diese bestehen bei allen betrachteten Mensakarten

nur aus Null-Bytes. Die Bytes 40 und 46 sind immer auf 0x0f gesetzt (vermutlich handelt es sich auch hier um einen 6 Byte Eintrag mit anschließender 2 Byte Prüfsumme).

5.5 Angriffsvektoren

Für jedermann, der auf oben beschriebene oder andere Art in Kenntnis des Datenformats und des Schlüssels A gelangt ist, lassen sich folgende Angriffe durchführen, die alle auf Manipulation des Karteninhalts aufbauen. Es wird davon ausgegangen, dass es kein Backend-system gibt das beispielsweise Schattenkonten führt oder eine Form der Betrugserkennung durchführt.

5.5.1 Replay-Angriff

Es ist möglich, eine Kopie der Karteninhalte zu einem bestimmten Zeitpunkt anzufertigen und dann nach einer Veränderung des Karteninhalts durch das System wieder zurückzuspielen. Damit hat die Karte wieder den Zustand vor der Änderung. Fertigt man zum Beispiel vor einem Bezahlvorgang eine Kopie an, so kann dies ausgenutzt werden, um den auf der Karte gespeicherten Betrag wieder auf den Betrag vor der Abbuchung zu setzen. Die Abbuchung würde dem Angreifer so keine Kosten verursachen.

5.5.2 Mitarbeiterkarte mit günstigeren Preisen

Indem die "Category-ID" und die "Tariff-ID" einer weißen Karte auf 0x01 bzw. 0x0100 gesetzt werden, ist es möglich, mit einer weißen Karte zu günstigeren Studentenpreisen zu speisen. Eine optische Überprüfung der Kartenfarbe ist ausschliessbar, da es Usus, ist seine Karte im geschlossenen Geldbeutel auf den Leser der Kasse zu legen, so dass die Karte nicht sichtbar ist.

5.5.3 Kartenvervielfältigung

Indem man eine Karte kopiert und anschließend auf eine andere MIFARE Karte überträgt, ist es möglich, Klone einer Mensakarte zu erstellen. Für den Geldbetrag auf diesen Klonen wurde keine Einzahlung in das System geleistet. Es ist jedoch möglich mit diesen zu zahlen.

5.5.4 Rein lesbare Karten

Indem man die Zugriffs-Bits entsprechend modifiziert, ist es möglich, die Karte rein lesbar zu machen. Sie hätte dann immer denselben Betrag, eine Abbuchung bliebe folgenlos.

5.5.5 Aufstellen eigener Aufladeautomaten

Da die Software, die auf den Aufladeautomaten läuft, via Google zu finden ist, wäre es möglich, ein Gehäuse herzustellen, das dem der Mensaautomaten gleicht und darin einen Laptop oder ein ähnliches Gerät mit der Software zu betreiben. An dieser Attrappe kann dann einfach Geld angenommen werden und den gutgläubig ihr Guthaben aufladenden Personen dieses auch gutgeschrieben werden. Es ist jedoch nicht zu erwarten, dass ein Angreifer dem Studentenwerk auch das eingenommene Geld übergeben würde. Auch wenn die Software nicht einfach zu finden wäre, ließe sich mit etwas Aufwand eine sich gleich verhaltende Software mit gleichem Aussehen schreiben, um den Angriff durchzuführen.

5.5.6 Denial of Service durch "Massenmanipulation"

Es ist möglich, einen Denial-of-Service Angriff auf das System durchzuführen. Da alle Mensakarten mit demselben Schlüssel geschrieben werden können und das Schreiben der Karteninhalte über Funk möglich ist, ist es möglich, den Betrag auf einer Karte im Besitz einer anderen Person zu verändern, ohne dass diese Person das bemerken kann. Es ist zum Beispiel denkbar, mit entsprechenden Antennen ausgestatte RFID-Leser an Hörsaaleingängen oder dem Eingang zur Mensa verdeckt anzubringen und jeder vorbeikommenden Mensakarte (in der Hosentasche eines Unbeteiligten) das aktuelle Guthaben auf beispielsweise 500€ zu setzen. Eine solche Manipulation in großem Maßstab würde mit Sicherheit den Argwohn der Kassiererinnen erregen und sie würden vermutlich die Zahlungsannahme verweigern. Wenn eine genügend große Anzahl solcher manipulierten Karten auftauchten, wird der Betrieb der Mensa sicherlich gestört werden bzw. zum Erliegen kommen.

5.6 Mensatool

Um zu verstehen, wie die Mensakarten genau funktionieren, Testreihen zu automatisieren und Ergebnisse anschaulich präsentieren zu können, habe ich ein kleines Tool unter dem Arbeitstitel "mensatool" entwickelt. Es beinhaltet wahlweise eine kommandozeilenorientierte Schnittstelle (Abbildung 5.7) oder eine graphische Benutzeroberfläche (Abbildung 5.8). Die Operationen, die die beiden Schnittstellen anbieten, sind unterschiedlich, aber nicht disjunkt.

5.7 Backendsystem

Aus einem Dokument[3] der "tl1 GmbH" geht hervor, dass die Aufladeautomaten mit einer Backendsoftware kommunizieren können und so die Möglichkeit

[3] http://80.237.169.16/DownloadPub/TlmChipkarten.pdf

```
[14:05:49][spjsschl@faui03a:~/MIFARE/crapto1]$ ./mensatool -h
Usage: ./mensatool parameters
with the following parameters:
                -c sniffdatafile        Crack key from sniffed data in
sniffdatafile
                -k xxxxxxxxxxxx         Use key k supplied as 12 hexdigits
                -g                      Get current value
                -s intvalue             Set value to intvalue
                -d                      dump mensacardsectors
                -h                      Display this text
```

Abbildung 5.7: Hilfetext der Mensatool Komandozeilenschnittstelle

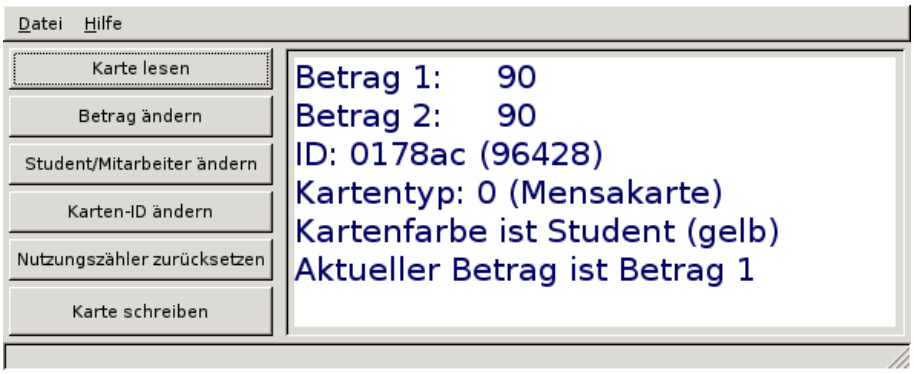

Abbildung 5.8: Graphische Benutzeroberfläche des Mensatools

geschaffen wird, Informationen über Buchungsvorgänge oder andere Änderungen auf den Karten, die nicht vom System selbst vorgenommen wurden, zu sammeln und auszuwerten um Betrugsfälle zu erkennen. In dem Dokument findet sich unter dem Stichwort "Datenhaltung" folgender Satz: "Alle Daten werden als Einzeltransaktionen in der Datenbank Oracle® gespeichert und bleiben dort min. 10 Jahre erhalten.". Weiterhin findet sich unter "Funktionalität im Überblick" der Stichpunkt "Kartensalden-Verfolgung, Aufspüren auffälliger Transaktionen".

Viele der oben beschriebenen Angriffe wären mit einem Backendsystem erkennbar, verhinderbar oder verfolgbar. Es ist nicht ganz klar, ob ein Backendsystem in der FAU eingesetzt wird. Dafür spricht die Tatsache, dass ein solches Backendsystem vom Hersteller angeboten wird. Ein strenges Schattenkonto kann von dem Backendsystem nicht geführt werden, da es bereits erwähnte Getränkeautomaten gibt, die nicht online sind und damit eine Transaktion nicht an ein Backendsystem zur anschliessenden Buchung auf dem Schattenkonto melden können. Es kann also Fälle geben, in denen ein Schattenkontowert von dem

Kartenwert abweichen würde und der Kartenwert der einzig richtige wäre. Daran ansetzend kann man sich leicht Angriffe ausdenken, die auch trotz Backendsystem erfolreich durchführbar wären.

Ob das oben genannte Backendsystem auch in der FAU im Einsatz ist, ob Schattenkonten geführt werden, ob irgendeine Form der automatischen oder manuellen Betrugserkennung auf einer eventuell erhobenen Datenbasis vorgenommen wird und daher auch, in welchen Grenzen bzw. ob überhaupt, etliche der oben skizzierten Angriffe möglich sind, entzieht sich meiner Kenntnis und wäre durch weitere Arbeiten zu untersuchen.

Kapitel 6

Kritische Würdigung

Die Forschungsarbeiten vor allem im letzten Jahr zu MIFARE haben gezeigt, dass es eine Reihe Schwachstellen gibt, die von einem guten Sicherheitsteam im Vorfeld hätten erkannt werden müssen. Durch die Verbreitung der Karten in den letzten Jahren hat ein Interesse an deren Funktionsweise eingesetzt, das dazu geführt hat, dass in kurzer Zeit von einer geheimgehaltenen und vorgeblich sicheren Technologie nicht viel geblieben ist. Es zeigt sich am Beispiel MIFARE wieder einmal sehr klar, dass security by obscurity nicht funktioniert - im Gegenteil. Die Kosten, die entstehen, wenn ein vermeintlich sicheres System nach Jahren ausgetauscht werden muss, sind vermutlich höher als von Anfang an ein sicheres System, das der Kritik von Sicherheitsexperten standhält und sich ihrer bedient, zu entwickeln. Ein offen entstandenes System hat es auch leichter, das Vertrauen der Leute zu gewinnen, die es einsetzen, da es für jedermann möglich ist, nachzuvollziehen, wie es funktioniert.

Aus einem neuen Paper von geht hervor, dass es möglich ist, sämtliche Schlüssel einer MIFARE-Classic Karte effizient zu berechnen, wenn ein einziger Schlüssel bekannt ist[1]. Damit entfällt für viele Systeme der Aufwand, den geheimen Schlüssel aus einer mühsam mitgeschnittenen Kommunikation zu errechnen, nämlich in all den Systemen wo auf der Karte noch Sektoren mit bekannten Werksschlüsseln verbleiben. Dies ist bei fast allen mir bekannten Systemen der Fall. Damit sinkt die Schwelle weiter, die ein Angreifer überwinden muss, um ein MIFARE-basiertes System anzugreifen. Dies ist eine weitere Schwachstelle in den kryptographischen Algorithmen und deren Implementierung die für MIFARE-Classic verwendet wurden. Wie eingangs erwähnt wird hier mittlerweile nachgebessert, so dass zu hoffen bliebt, dass nachfolgenden MIFARE-Systemen eine längere Lebensspanne beschieden ist.

Die vor allem in 4.2 erwähnten Schwachstellen abseits der MIFARE-Classic Karte selbst lassen darauf schliessen, dass auch ein kryptographisch sicheres System oftmals andere Schwachstellen in seiner Gesamtheit bietet, die zu einer Kom-

[1] http://www.proxmark.org/documents/mifare_weakness.pdf

promittierung des Systems führen können. Sicherheitsziele durchzusetzen ist in der Praxis nie ein eindimensionaler Belang und kann nicht allein durch sichere kryptographische Algorithmen erreicht werden. Zahlreiche Querschnittsbelange müssen bedacht und richtig umgesetzt werden, um im Abschluss ein sicheres, zuverlässiges, seine Aufgaben erfüllendes, dem Benutzer dienendes und vertrauenswürdiges System zu schaffen.

Jede scheinbare Sicherheit ist im Gegenteil eben nur scheinbar. Sie trügt, wenn man sich auf sie verlässt und führt dazu, dass das Gegenteil von dem eintritt, was das Ziel war. Statt ein Sicherheitsproblem zu beseitigen, bleibt es offen und wiegt die Verantwortlichen in vermeintlicher Sicherheit, was deren Wachsamkeit mindert.

Vor allem Datenschutz und Privacyprobleme lassen sich auch mit sicheren kryptographischen Methoden nicht lösen, da sie im Wesen der RFID-Karten als funkende Karten an sich, oder in der mit solchen Systemen einhergehenden Datenhaltung (im Backend oder auf den Karten selbst) verwurzelt sind.

Kapitel 7

Ausblick

Im Umfeld um die MIFARE-Karten und allgemein die RFID-Technik ist noch viel weitere Arbeit möglich. Im Anschluss an diese Arbeit würde es sich anbieten, das Backendsystem der Mensa genau zu untersuchen: Welche Formen der Manipulation fallen auf, welche bleiben unentdeckt, welche können eventuell nicht entdeckt werden?

Weiterhin könnte man die MIFARE-Karten, die als Kopierkarte in der Universität eingesetzt werden, untersuchen. Stichproben haben ergeben, dass die Kopiergeräte anscheinend nicht über eine Aussenanbindung verfügen. Die Kopiergeräte sind im Gegensatz zu den Aufladeautomaten der Mensa aber Einzugsgeräte, deren Einzugsslot so eng ist, dass es kaum gelingt, eine Antenne für sinnvolle Sniffvorgänge einzubringen. Hier könnte es helfen, eine Kopierkarte abzuschleifen und den damit gewonnenen Platz für eine Antenne zu nutzen. Abzuklären wäre dann noch, ob die Antenne so noch ein brauchbares Signal empfängt. Die Aufladeautomaten für die Kopierkarten stehen zumindest an der technischen Fakultät im Sichtbereich von Überwachungskameras.

Interessant wäre es auch zu testen, wieweit die passiven MIFARE-Karten ausgelesen werden können, wenn man eine entsprechende darauf ausgelegte Antenne baut. Damit einhergehend kann auch untersucht werden, wie eine ideale Antenne für Relayangriffe aussieht und wie zuverlässig ein solcher Angriff ohne weitere Hilfsmittel durchgeführt werden könnte.

Der Replay-angriff auf die Moskauer Metrofahrkarten ist bisher auch ausschliesslich theoretischer Natur und dessen Durchführbarkeit könnte noch praktisch untersucht werden. Sicher ergeben sich noch zahlreiche ähnlich gelagerte Angriffsmöglichkeiten in Parkhäusern, bei Skipässen oder anderen Fahrkartensystemen, die hier nicht betrachtet worden sind.

Literaturverzeichnis

[1] Henryk Plötz, *MIFARE Classic - eine Analyse der Implementierung*. Diplomarbeit, Humboldt-Universität zu Berlin, 2008.

[2] Flavio D. Garcia e.a., *Dismantling MIFARE Classic*. Radboud University Nijmegen, 2008.

[3] Karsten Nohl, Henryk Plötz, *MIFARE, little security, despite obscurity*. Präsentation auf der 24C3 des Chaos Computer Clubs in Berlin, 2008.

[4] Auguste Kerckhoffs, *La cryptographie militaire*. Journal des Sciences Militaires IX, 5-38, 1883.

[5] Karsten Nohl, Henryk Plötz, D. Evans, starbug *Reverse-engineering a cryptographic RFID tag*. USENIX Security 2008, 2008.

[6] Lukas Grunwald, *New Attacks against RFID-Systems*. Black Hat Briefings USA, 2006.

Abbildungsverzeichnis