

Technische Spezifikation der RFID-Umgebung  
zur interoperablen Authentifizierung v1.0  
**Draft, durch Tests zu bestätigen**

Johannes Bauer

Johannes.Bauer@bosch-si.com

Bosch Software Innovations GmbH

6. Juni 2011

# Inhaltsverzeichnis

<b>1</b>	<b>Dokumentfokus</b>	<b>6</b>
1.1	Nomenklatur . . . . .	6
<b>2</b>	<b>Hardware</b>	<b>6</b>
2.1	RFID-Standard . . . . .	6
2.2	SAM-Module . . . . .	6
2.3	Schlüsselableitung . . . . .	7
2.4	Datenmodell . . . . .	7
2.5	Möglichkeiten . . . . .	7
2.6	Minimalanforderung . . . . .	7
2.7	Kryptografische Grundvoraussetzung . . . . .	8
2.8	Einbettung in Zertifikate . . . . .	8
2.9	Reaktion auf kryptografische Schwachstellen . . . . .	9
2.10	Patentsituation von ECC . . . . .	9
<b>3</b>	<b>Implementierungsdetails</b>	<b>10</b>
3.1	Repräsentation von $I_R$ . . . . .	10
3.2	Schlüsselerzeugung . . . . .	10
3.3	Erzeugung des Zertifikats . . . . .	11
3.4	Erzeugung von Karten . . . . .	12
3.5	Aufbringung eines Verfalldatums . . . . .	12
3.6	Aufbringung des Datenblocks . . . . .	14
3.7	Alternativer Datenblock . . . . .	14
3.8	Herstellerspezifischer Datenblock . . . . .	15
3.9	Übersicht des RFID-Layouts . . . . .	16
3.10	Verteilung der öffentlichen Schlüssel . . . . .	16
<b>4</b>	<b>Sicherheitsanalyse</b>	<b>17</b>
4.1	Fälschen von Karten . . . . .	17
4.2	Kopie von Karten . . . . .	18
4.3	Emulations-Angriff . . . . .	18
<b>5</b>	<b>Wahl des symmetrischen Schlüssels</b>	<b>18</b>
5.1	Alternativen . . . . .	19
5.2	Plausibilität des Angriffsszenarios . . . . .	21
5.3	Empfehlung . . . . .	22
<b>6</b>	<b>Ausblick</b>	<b>22</b>
6.1	Notwendigkeit einer Migration . . . . .	22
6.2	Migrationsstrategie . . . . .	22

<b>Literatur</b>	<b>26</b>
<b>Stichwortverzeichnis</b>	<b>29</b>

## Revisionen

0.01	2010-06-17	Initiale Revision
<hr/>		
0.02	2010-06-18	Anmerkungen von Prof. Dr. Ruland (Universität Siegen) eingepflegt: <ul style="list-style-type: none"><li>• Bezeichner <math>I_C</math> und <math>I_R</math> berichtigt</li></ul>
<hr/>		
0.03	kein Release	Anmerkungen von Prof. Dr. Ruland (Universität Siegen) eingepflegt: <ul style="list-style-type: none"><li>• Verfahren zur Schlüsselableitung für Kartenschlüssel gefordert</li><li>• Verfahren zur Schlüsselverteilung zum Risikosplitting nahegelegt</li></ul>
<hr/>		
0.04	kein Release	Anmerkungen von Anton Schmitt (Siemens) eingepflegt: <ul style="list-style-type: none"><li>• Implikationen eines Wechsels des Kryptosystems näher erläutert</li><li>• Sicherung des Keycaches explizit erwähnt</li><li>• Differenziertere Betrachtung der Schwierigkeit des Emulationsangriffs</li><li>• Diverse kleine Verbesserungen</li></ul>
<hr/>		
0.05	2010-07-07	<ul style="list-style-type: none"><li>• Fordern der S/MIME Version 2</li><li>• Hinzufügen von empfohlenen Laufzeiten der Zertifikate</li><li>• Hinzufügen diverser Testvektoren</li></ul>
<hr/>		

---

0.06 2010-07-08      • Erweiterung des Abkürzungsverzeichnisses

---

Anmerkungen von Jörg Röhlen (Regio IT Aachen) eingepflegt:

0.07 2010-11-11      • Repräsentation der DESFire-ID geklärt  
• Application- und File-ID auf der DESFire-Karte definiert  
• Migrationsstrategie erläutert

---

0.08 2010-11-17      • Diskussion um symmetrischen Schlüssel  $K_S$  eingefügt

---

Anmerkungen von Håkan Källberg (Simulina) eingepflegt:

0.09 2010-11-17      • Verfahren zur Prüfung der Karten-Signatur berichtigt und eindeutig formuliert

---

Anmerkungen von Steffen Fries (Siemens) eingepflegt:

0.10 2010-11-22      • Erweiterung des Abkürzungsverzeichnisses  
• Klärung zur Patentsituation um ECC eingefügt  
• Definition eines optionalen Feldes im Subject der X.509-Zertifikate  
• Hinzufügen des RFID-Emulationskit der TU Graz

Anmerkungen von Håkan Källberg (Simulina) eingepflegt:

• Verwendung eines neutralen Begriffs für Kartenherausgeber

---

---

		Anmerkungen von Jörg Röhlen (Regio IT Aachen) eingepflegt:
0.11	2011-01-13	<ul style="list-style-type: none"> <li>• Möglichkeit der Speicherung einer unsignierten ID auf der RFID-Karte hinzugefügt</li> </ul>
		<ul style="list-style-type: none"> <li>• Key Diversification im Detail beschrieben</li> <li>• Appendix mit Beispielen für Zugriff auf die RFID-Karte hinzugefügt</li> </ul>
0.12	2011-01-31	<ul style="list-style-type: none"> <li>• Verwendete Schlüssel im Detail beschrieben</li> <li>• Verwendung eines Verfalldatums für Karten</li> <li>• Verwendung einer Vendor-Specific File-ID hinzugefügt</li> </ul>
1.0	2011-06-06	<ul style="list-style-type: none"> <li>• Übernahme des Entwurfs als aktuell gültige Version, durch Modellregionen im Test zu überprüfen</li> </ul>

---

# 1 Dokumentfokus

Dieses Dokument erläutert die Hardware- und Softwaregegebenheiten, die für eine interoperable RFID-Lösung im Rahmen der IKT Modellregionen sowie Ladesäulenherstellern von Drittanbietern notwendig sind. Hierfür wird eine kurze Erläuterung der technischen RFID-Gegebenheiten präsentiert um nachfolgend auf das darauf aufsetzende Datenmodell näher einzugehen. Für das Verständnis hilfreich ist Kenntnis der Systemsicht des Verfahrens, welche unter den Arbeitspapieren der TaskForce AIEV AK RFID zu finden ist. [11]

## 1.1 Nomenklatur

Für Schlüssel symmetrischer Kryptosysteme wird im folgenden  $K$  mit einem Index, der die Verwendung anzeigt verwendet. Für asymmetrische Kryptosysteme wird  $K_{\text{Pub}}$  und  $K_{\text{Priv}}$  verwendet, welche als Schlüsselpaar nur zusammen sinnvoll sind.  $K_{\text{Pub}}$  bezeichnet hier den öffentlichen (public) und  $K_{\text{Priv}}$  den geheimen (private) Schlüssel. Eine Verwendung von Indizes zeigt hier den Besitzer des geheimen Schlüssels (und somit den Erzeuger des Schlüsselpaars) an, z.B.  $K_{\text{Priv}_A}$ .

# 2 Hardware

## 2.1 RFID-Standard

Dem System liegt der RFID-Standard Mifare DESFire EV1 (früher unter dem Namen DESFire8 bekannt) zu Grunde. Dieses System setzt zur Kommunikation zwischen Lesegerät und RFID-Karte einen symmetrischen AES-Schlüssel von 128 Bit (16 Byte) Länge ein. Im Folgenden wird dieser Schlüssel nur noch mit *AES-Key* oder  $K_S$  bezeichnet. Jedes DESFire EV1 RFID-Tag verfügt über einen 7 Byte langen, eindeutigen Identifier, der bei Herstellung der Karte aufgebracht wird und nachträglich nicht mehr verändert werden kann. Dieser Identifier wird im Folgenden als *RFID-ID* oder  $I_R$  bezeichnet. Wie diese Seriennummer innerhalb des RFID-Konzepts repräsentiert wird, ist in 3.1 näher erläutert.

## 2.2 SAM-Module

Einige RFID-Lesegeräte bieten eine Hardware-Schnittstelle und entsprechende Software-Unterstützung für sogenannte SAM-Module an. Die Bauform der SAM-Module entspricht der einer vom Mobiltelefon bekannten SIM-Karte. Auf den SAM-Modulen kann der AES-Key  $K_S$  hinterlegt werden und ermöglicht es dem angeschlossenen RFID-Lesegerät so, Daten mit  $K_S$  zu ver- und entschlüsseln.

SAM-Module bieten den Vorteil, dass bei Diebstahl eines Moduls der geheime AES-Key nicht bekannt wird und repliziert werden kann, da das SAM-Modul als Schnittstelle zum Leser zwar mit Hilfe des einmal eingebrachten Schlüssels Daten ver- und entschlüsseln kann, der Schlüssel später allerdings nicht mehr auslesbar ist. Technologisch ähnlich lösen viele Hersteller das Problem der Schlüsselverwahrung mit Sonderlösungen, bei denen die AES-Schlüssel in geschütztem Speicher auf einer Hardwareeinheit hinterlegt werden.

### 2.3 Schlüsselableitung

Um den  $K_S$  nicht auf jedem RFID-Tag zu hinterlegen, muss ein Verfahren gewählt werden, welches aus einem *Master-Key*  $K_M$  zusammen mit  $I_R$  einen kartenabhängigen Schlüssel  $K_S$  diversifiziert. Hierfür ist sinnvollerweise ein Verfahren zu wählen, welches auch von den in 2.2 beschriebenen SAM-Modulen unterstützt wird. Deswegen wird hierfür jenes Verfahren verwendet, welches NXP in der offiziellen, öffentlichen Key Diversification Application Note beschreibt. [21] Essentiell kommt für dieses Verfahren AES-CMAC zum Einsatz, welches in NIST Special Publication 800-38B beziehungsweise RFC4493 beschrieben ist. [17][12] In der RFC findet sich darüber hinaus eine Referenzimplementierung.

### 2.4 Datenmodell

### 2.5 Möglichkeiten

Die in Abschnitt 2.1 beschriebene RFID-Kartentechnologie bietet bestimmte Möglichkeiten an und hat diverse Einschränkungen. Beides muss bei der Entscheidungsfindung berücksichtigt werden. Die Adressierung von Datenblöcken geschieht bei Mifare DESFire EV1 – anders als bei den meisten anderen RFID-Karten – nicht sektorweise, sondern stellt eine Art rudimentäre Dateisystemabstraktion bereit. [20] Im DESFire-Kontext werden die Bezeichnungen „*Application*“ für eine Art Verzeichnis und „*Files*“ für eine Art Dateien innerhalb dieses Verzeichnisses verwendet. Applications und Files werden jeweils durch ihre Nummer referenziert. Eine DESFire-Application kann bis zu 32 Files pro Application besitzen und bis zu 28 Applications auf der Karte insgesamt verwalten. Zusätzlich besteht eine Größenbeschränkung auf 2, 4 oder 8 kB pro RFID-Karte.

### 2.6 Minimalanforderung

Das Modell zur interoperablen Authentifizierung sieht vor, dass auf den RFID-Karten eine logische ID aufgebracht wird, welche von der Arbeitsgruppe Nummernkreise näher definiert wird. In verschiedenen Kontexten wird diese ID oft

unterschiedlich bezeichnet. Im Folgenden wird diese ID *Contract-ID* oder  $I_C$  genannt. Diese Contract-ID soll zusätzlich digital signiert werden und diese Signatur ebenfalls auf der RFID-Karte aufgebracht werden. Da die Signatur nicht einfach von einer Karte auf eine andere kopiert werden können soll ist es notwendig, dass die RFID-ID in die Signatur miteinbezogen wird.

Die Sicherheit des Verfahrens hängt also grundsätzlich von zwei Punkten ab, von denen jeder alleine ausreichend ist, die Gesamtsicherheit des Verfahrens zu wahren: Zum Einen ist dies die Geheimhaltung von  $K_S$ , welche nicht unter allen Umständen gewährleistet werden kann. Zum Anderen ist dies die Nichtkopierbarkeit der RFID-Karten und damit die Eindeutigkeit von  $I_R$ . Auch hierauf ist, wie später in Kapitel 4.3 gezeigt wird, ein Angriff möglich, dessen Risiko abgewogen werden muss.

## 2.7 Kryptografische Grundvoraussetzung

Durch die Ressourcenbeschränkungen, die RFID-Karten mit sich bringen, ist es sinnvoll, die aufgebrachten digitalen Signaturen so kurz wie möglich zu halten. Hierfür ist der ECC-Algorithmus gut geeignet. [1][3] Er bietet trotz kurzer Schlüssel ein vergleichsweise hohes Sicherheitsniveau. [2] Laut BSI-Empfehlung sollten EC-Schlüssel mindestens 200 Bit lang sein (entspricht etwa einem RSA-Äquivalent von 2048 Bit). Das EC-Verfahren setzt voraus, dass die elliptischen Kurven, auf denen die Schlüssel basieren, jedem Teilnehmer bekannt ist. Hierfür gibt es von der NIST Empfehlungen für Kurven unterschiedlicher Schlüssellängen, die auf FIPS 186-3 basieren. [7] Für das vorliegende Szenario wird secp224r1 empfohlen, welche auch in RFC5480 näher beschrieben ist. [23]

## 2.8 Einbettung in Zertifikate

Durch eine Verwendung von reinen Schlüsseln wäre ein proprietäres System geschaffen, welches in der Flexibilität und Kompatibilität weit gegenüber aktuellen technischen Möglichkeiten hintenanstünde. Da die Auswirkung auf die Gesamtlänge der Schlüssel nicht dramatisch ist, sich die Flexibilität und Offenheit des Systems durch eine Verwendung von X.509-Zertifikaten erheblich verbessert, wird eine Verwendung derer vorgeschlagen. Im ersten Schritt sollte es sich lediglich um selbst-signierte X.509-Zertifikate (auch bekannt als Root-Zertifikate) handeln, die an einer zentralen Stelle zwischen den Ladesäulenherstellern ausgetauscht werden. Eine derartige Verwendung hat mehrere Vorteile:

1. Die Algorithmen zur Verschlüsselung sind austauschbar, falls sich diese z.B. durch neue kryptoanalytische Verfahren als unsicher herausstellen.

2. Die Algorithmen zur Erzeugung eines Hashwertes, welcher unmittelbar mit der digitalen Signatur zusammenhängt, sind ebenfalls austauschbar.
3. Die Schlüssellängen für Verschlüsselung und Signatur sind parametrierbar.
4. Die Migration von einer flachen Hierarchie (in der es nur Root-Zertifikate gibt) hin zu einer PKI ist problemlos möglich.
5. X.509-Zertifikate sind Stand der Technik, haben einen sehr hohen Verbreitungsgrad (zum Beispiel bei https), sind patentfrei und es gibt zahlreiche gemeinfreie Werkzeuge um diese kostenlos zu erzeugen, verändern und um damit Daten zu signieren oder zu verschlüsseln.

Zum interoperablen Verwendung sollten Signaturen in einer S/MIME-Hülle (Version 2) eingebettet.

## **2.9 Reaktion auf kryptografische Schwachstellen**

Sollte in einem der angewendeten kryptografischen Verfahren eine Schwachstelle entdeckt werden, so müssen diejenigen Zertifikate, die dieses Verfahren benutzen, ausgetauscht werden – im Zweifelsfall sind dies alle ausgestellten Zertifikate. Dies kann im Rahmen eines turnusmäßigen Zertifikatsaustauschs, der ohnehin vorgesehen werden sollte, erfolgen (also zeitverzögert) oder muss unmittelbar erfolgen. Technisch ist dies unproblematisch, ein solcher Austausch kann allerdings mit einigem organisatorischen Aufwand verbunden sein. Ob die Zertifikate unmittelbar oder zeitverzögert ausgetauscht werden, hängt von der Kritikalität der assoziierten Sicherheitsschwachstelle ab. Gilt ein Verfahren als trivial gebrochen müssen die Zertifikate selbstverständlich sofort ausgetauscht werden. Ein Beispiel für eine derartige Schwäche ist der Angriff auf WEP, ein Verfahren, das im Wireless-Umfeld mit dem Algorithmus RC4 verwendet wurde. [6] In der Regel werden kryptografische Verfahren jedoch nicht derartig gebrochen, sondern die Komplexität des Angriffs verringert sich mit neuen Forschungsergebnissen um Größenordnungen, wie wir dies momentan beispielsweise mit den Angriffen auf die kryptografische Hashfunktion MD5 sehen [13]. Eine derartige Schwachstelle war auch bei RC4 schon etwa 8 Jahre bekannt, bevor der Angriff von Tews et al den Algorithmus letztendlich zur Farce werden ließ. [16]

## **2.10 Patentsituation von ECC**

Die momentane Patentsituation von ECC führt häufig zu Verunsicherung und Zurückhaltung, was deren Verwendung angeht. Insbesondere Certicom hat auf

bestimmte effiziente Verfahren, die Manipulationen auf ECC-Parametern durchführen, eine erhebliche Anzahl an Patenten. Diese schützen jedoch lediglich diese bestimmten Implementierungen – das Verfahren der ECC selbst ist patentfrei. Es ist sehr wichtig, diesen Unterschied deutlich hervorzuheben. Eine wichtige Schlussfolgerung daraus ist, dass es durchaus patentfreie Implementierungen von ECC gibt, wie sie beispielsweise in OpenSSL implementiert wurden. [14][5]

### 3 Implementierungsdetails

Die folgenden Absätze zeigen detailliert, wie eine Schlüssel- und Signaturerzeugung so zu erfolgen hat, dass Interoperabilität gewährleistet ist. Es werden jeweils konkrete Beispiele angegeben, welche sich jeweils auf Funktionalität des gemeinfreien OpenSSL-Toolkits beziehen.

#### 3.1 Repräsentation von $I_R$

Um die RFID-ID  $I_R$  innerhalb des Authentifizierungsschemas immer auf dieselbe Art und Weise auf die Datenblöcke aufzubringen, muss definiert werden, wie dieser 7 Byte lange Datenblock in ASCII-Form repräsentiert wird. Hierzu wird festgelegt, dass die in der DESFire EV1 Spezifikation (siehe [20]) festgelegten Bytes  $UID_0 \dots UID_6$  in eben dieser Reihenfolge in hexadezimaler Form konkateniert werden. Hierbei wird jedes Byte der UID auf eine zwei Zeichen lange Hexadezimalzahl mit führender Null abgebildet. Für die Zahlen 10-15 (a-f) sind Kleinbuchstaben zu verwenden. Die Endrepräsentation muss also dem regulären Ausdruck  $[0-9a-f]\{14\}$  genügen.

#### 3.2 Schlüsselerzeugung

Zunächst muss von jedem Ladesäulenhersteller ein Schlüsselpaar  $K_{Pub} / K_{Priv}$  mit den oben genannten Parametern erzeugt werden:

```
$ openssl ecparam -name secp224r1 -genkey -out Region.key
```

Welches eine Datei `Region.key` erzeugt, in der das Schlüsselpaar beinhaltet ist. Diese Datei muss von dem Ladesäulenhersteller sicher verwahrt werden, bei bekannt werden ist die Sicherheit des Kryptosystems kompromittiert.

### 3.3 Erzeugung des Zertifikats

Um das Zertifikat aus dem Schlüsselpaar zu erzeugen müssen zunächst dem Schlüssel Informationen über den Besitzer hinzugefügt werden. Dies erstellt ein CSR<sup>1</sup>:

```
$ openssl req -new -utf8 -key Region.key
  -subj '/C=DE/ST=BW/L=Immenstaad
        /CN=MeRegioMobil Modellregion Bodensee
        /OU=Johannes Bauer'
  -out Region.csr
```

Folgende Felder des X.509 Certificate Subject sind hier verwendet:

- **C (Country)**, zwingend erforderlich: Länderkennung nach ISO 3166
- **ST (State)**, zwingend erforderlich, falls anwendbar: Länderuntergliederung nach ISO 3166-2 (für Deutschland ist dies beispielsweise die Kennzeichnung des Bundeslandes)
- **L (Locality)**, zwingend erforderlich: Standort des verantwortlichen Firmensitzes
- **CN (Common Name)**, zwingend erforderlich: Name des Ladesäulenherstellers oder der Modellregion
- **OU (Organizational Unit)**, zwingend erforderlich: Name der für die Schlüsselerzeugung verantwortlichen juristischen Einheit oder Person

Die Felder C, ST, L, CN sowie OU sind alle zwingend erforderlich und dürfen im Zertifikats-Subject nur ein einziges Mal auftreten. Konforme Implementierungen müssen Zertifikats-Subjects, die eines der Felder mehrfach enthalten oder eines der Felder kein einziges Mal enthalten als ungültig zurückweisen. Zusätzliche Felder, die möglicherweise vom Kartenherausgeber aufgebrachte, benutzerdefinierte Informationen enthalten, sind von konformen Implementierungen zu ignorieren, wenn diese ihr nicht bekannt sind.

Nach der Erstellung eines CSR kann nun das CSR durch den geheimen Schlüssel signiert werden und wird dadurch zum vollwertigen Zertifikat:

---

<sup>1</sup>Certificate Signing Request

```
$ openssl x509 -req -days 730 -in Region.csr
    -signkey Region.key -set_serial 12345
    -out Region.crt
```

Die Seriennummer des Zertifikats sollte mit 1 beginnen und lückenlos sequentiell fortgeführt werden. Die Laufzeit des Zertifikats sollte nicht länger als 2 Jahre sein.

### 3.4 Erzeugung von Karten

Die S/MIME-Signatur muss zur Erzeugung gültiger Karten über ein Datum erzeugt werden, welches sowohl  $I_C$  als auch  $I_R$  enthält. Dies geschieht durch einfache Konkatenation von  $I_C$  an  $I_R$  getrennt durch einen Bindestrich. Abschließend darf kein Zeilenterminator in dem Datum vorkommen (kein LF = 0x0a oder CRLF = 0x0d0a). Zunächst muss hierfür  $I_R$  aus der RFID-Karte ausgelesen werden.  $I_R$  wird dann in hexadezimaler Form wie in 3.1 dargestellt (POSIX Format Specifier %014x). Wenn als Beispiel  $I_C = \text{DE-BO-123456}$  und  $I_R = \text{0xdeadbeefc0ffee}$ , dann kann der Ladesäulenhersteller die Signatur wie folgt erzeugen:

```
$ echo -n DE-BO-123456-deadbeefc0ffee
    | openssl smime -sign -signer Region.crt
      -inkey Region.key
    | gzip >Region-deadbeefc0ffee.sig
```

Der Besitzer des privaten Schlüssels erzeugt also eine Signatur über das konkatenierte Datum und komprimiert dies anschließend mittels des in RFC1952 beschriebenen Algorithmus. [22] Die erzeugte Datei `Region-deadbeefc0ffee.sig` kann nun auf die RFID-Karte als Datenblock gelegt werden. Dieser komprimierte S/MIME Datenblock wird fortan als *Keyblob* bezeichnet.

### 3.5 Aufbringung eines Verfalldatums

Auf jeder RFID-Karte muss ein File mit der File-ID 0x00 (Standard Data File) der Größe 4 Bytes aufgebracht sein, in welchem das Verfalldatum der Karte in UTC hinterlegt ist. Eine RFID-Karte ohne aufgebrachtes Verfalldatum ist ungültig. Wenn eine unbegrenzte Haltbarkeit der RFID-Karte gewünscht ist, muss ein File

mit dem Inhalt `0xffffffff` hinterlegt werden. Soll ein Verfalldatum hinterlegt werden, ist dies wie folgt als 32-Bit-Zahl zu kodieren:

- Bits [31 : 20], 12 Bit – Jahr (Wert von 0 - 4095)
- Bits [19 : 16], 4 Bit – Monat (Wert von 1 - 12)
- Bits [15 : 11], 5 Bit – Tag (Wert von 1 - 31)
- Bits [10 : 0], 11 Bit – Minute (Wert von 0 - 1439)

**Beispiel:** Kodierung des 1997-08-29, 15:47

- $1997 \cdot 2^{20} = 2094006272$
- $8 \cdot 2^{16} = 524288$
- $29 \cdot 2^{11} = 59392$
- $(15 \cdot 60 + 47) \cdot 2^0 = 947 \cdot 2^0 = 947$
- Aufzubringendes Datum:  $2094006272 + 524288 + 59392 + 947 = 2094590899 = 0x7cd8ebb3$

**Beispiel:** Rückkodierung des Datenblocks `b3ebd87c`

- `b3 eb d8 7c` interpretiert als Little Endian kodierter `uint32`  
→  $0x7cd8ebb3 = 2094590899$
- Jahr:  $(2094590899 \& 0xffff0000) \cdot 2^{-20} = 1997$
- Monat:  $(2094590899 \& 0xf0000) \cdot 2^{-16} = 8$
- Tag:  $(2094590899 \& 0xf800) \cdot 2^{-11} = 29$
- Stunde:  $(2094590899 \& 0x7ff) \text{ div } 60 = 15$
- Minute:  $(2094590899 \& 0x7ff) \text{ mod } 60 = 47$

Diese Zahl ist im Little Endian-Format in der Datei zu codieren, d.h. das Byte an Offset 0 hat den höchsten Wert. In der Datei steht also, vom niedrigen Offset (0) zum höchsten Offset (3) der Binärstring `b3ebd87c` (hier angegeben in Hex-Codierung). In den Beispielen bezeichnet „&“ die bitweise UND-Operation, „div“ die ganzzahlige Division und „mod“ den Rest bei ganzzahliger Division.

Für die Interpretation des Datenblocks gelten zusammenfassend folgende Regeln, von denen die erste zutreffende die gewünschte Aktion auslöst:

1. Kein Verfalldatum vorhanden → Karte ungültig
2. Verfalldatum vorhanden, gleich  $0xffffffff$  → Signatur prüfen
3. Verfalldatum vorhanden, Datenblock ungültig (z.B. einzelner Wert außerhalb des angegebenen zulässigen Bereichs) → Karte ungültig
4. Verfalldatum vorhanden, ausgelesenes Datum  $\geq$  momentanes Datum (UTC) → Signatur prüfen
5. Sonst (ausgelesenes Datum  $<$  momentanes Datum (UTC) → Karte ungültig

### 3.6 Aufbringung des Datenblocks

Auf der DESFire EV1 RFID-Karte ist die Signatur als Standard-Datenblock geeigneter Größe aufzubringen. Hierzu wird die Application-ID 0x494b54 mit der File-ID 0x01 (Standard Data File) verwendet. Als Master-Key sowie Application-Key werden keine Nullschlüssel verwendet, sondern Schlüsselwerte, die unter den Teilnehmern in separatem Dokument verteilt werden. Zur Diskussion über alternativen bei der Wahl des symmetrischen Schlüssels sind nähere Erläuterungen in Abschnitt 5 zu finden. Da die Anzahl an Teilnehmern, denen dieser symmetrische AES-Key  $K_S$  bekannt ist möglicherweise unüberschaubar groß wird, ist nicht davon auszugehen, dass  $K_S$  über einen längeren Zeitraum geheim bleibt. Das System muss deshalb auch nach bekanntwerden von  $K_S$  ein möglichst hohes Sicherheitsniveau bieten. Leider wird durch die prinzipbedingte Art, mittels derer symmetrische RFID-Karten funktionieren, ein Emulations-Angriff wie in 4.3 beschrieben nach bekanntwerden von  $K_S$  nie verhindert werden können. Eine Migrationsstrategie auf eine neuere Technologie und damit gelöste Sicherheitsprobleme ist daher in 6 beschrieben.

### 3.7 Alternativer Datenblock

Um möglichst schnell ein interoperables Szenario mit minimalem Implementierungsaufwand zu schaffen, wird zusätzlich zur Application-ID 0x494b54 eine File-ID 0x02 (Standard Data File) vorgesehen. Diese soll einen kurzen Datenblock enthalten, der lediglich  $I_C$  im ASCII-Format enthält. Somit wird die Möglichkeit geschaffen, den physikalischen Zugang zur Karte (Nutzen der Crypto-Funktionalität) bereits zu testen, ohne dass die Spezifikation vollumfänglich implementiert werden muss. Das Nutzen der unsignierten  $I_C$  stellt einen gangbaren Zwischenschritt dar,

der voll migrationsfähig ist und innerhalb kürzester Zeit implementiert werden kann. Selbstverständlich bietet dieser Zwischenschritt nicht die Sicherheit, die das System insgesamt zur Verfügung stellt – die Sicherheit beruht hier vollständig auf dem Geheimnis  $K_S$ . RFID-Karten dürfen sowohl einen signierten Datenblock (wie in 3.6 beschrieben) als auch eine unsignierte  $I_C$  enthalten. Es ist dem Ladesäulenhersteller überlassen, welche der Daten er auswertet, jedoch sei darauf hingewiesen, dass nur eine Auswertung des signierten Datenblocks garantiert, dass  $I_C$ -Werte vom tatsächlich berechtigten Herausgeber auf der Karte aufgebracht wurden.

### **3.8 Herstellerspezifischer Datenblock**

Der Regelfall der Authentifizierung ist nicht Roaming, sondern die Identifikation eigener Kunden. Um dieses – recht häufig auftretende – Szenario zu beschleunigen, wird ein zusätzliches Standard Data File mit der File-ID 0x03 definiert. Diese Datei wird mit einem herstellereigenen Schlüssel aufgebracht und gibt so jedem Kartenherausgeber die Möglichkeit, ein Geheimnis zu hinterlegen, welches er selbst sehr effizient prüfen kann. Im einfachsten Fall kann hier lediglich die Kundennummer hinterlegt sein, es können in diesem Datenblock aber selbstverständlich auch andere Daten gespeichert werden. Ein nicht vorhandenes File 0x03 macht eine RFID-Karte nicht ungültig, diese herstellerspezifische Erweiterung ist vollständig optional. Auch ein nicht bekannter symmetrischer Schlüssel, um das File 0x03 zu lesen macht die RFID-Karte nicht ungültig. Falls das File nicht gelesen werden kann oder der Inhalt nicht interpretiert werden kann, ist es zu ignorieren.

### 3.9 Übersicht des RFID-Layouts

In Abbildung 1 wird noch einmal zusammengefasst, welche Schlüssel verwendet werden und welche Dateien in der Application hinterlegt werden müssen, um ein interoperables Szenario zu schaffen. Für Details zu den Schlüsseln und der Konfiguration der Karte ist der Appendix B zu konsultieren.

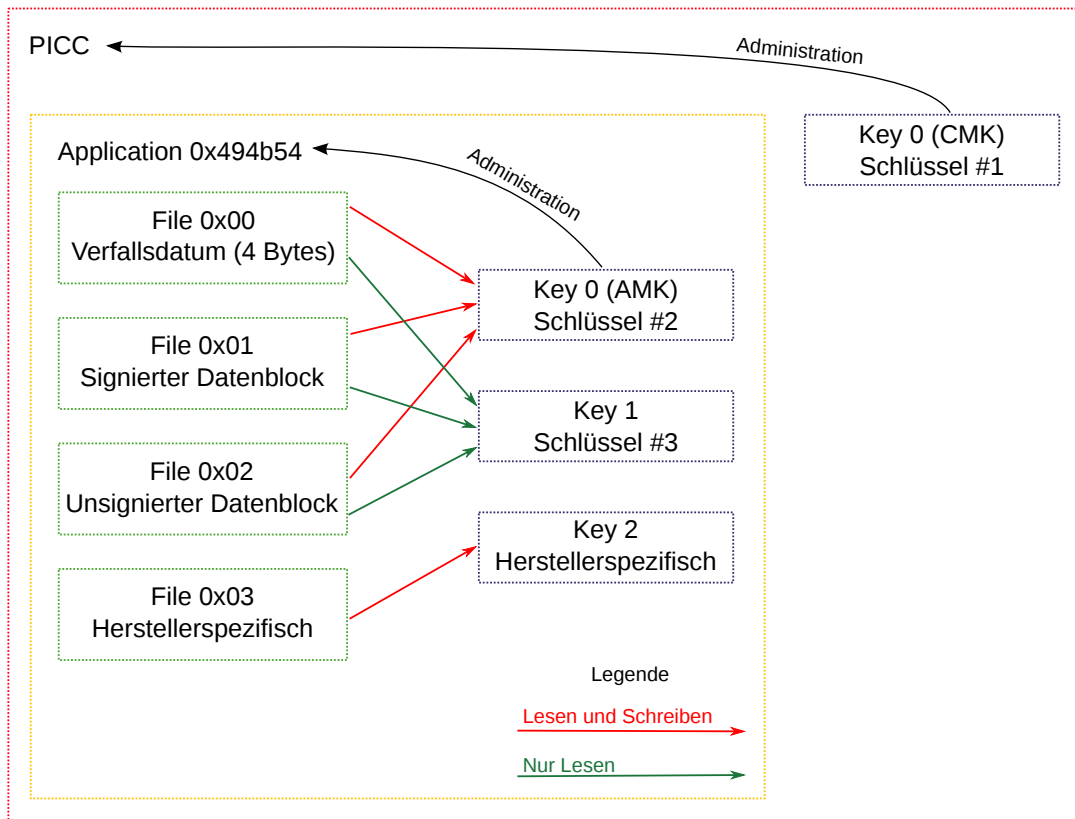


Abbildung 1: Zusammenfassendes Layout der RFID-Karte

### 3.10 Verteilung der öffentlichen Schlüssel

Die öffentlichen Schlüssel werden im Rahmen eines noch näher zu definierenden Verfahrens über einen Webserver als signiertes Archiv bereitgestellt, sodass sie von allen Ladesäulenherstellern abrufbar sind. Da dies für die erste Iteration noch nicht notwendig ist, da die Anzahl der Teilnehmer und Schlüssel gering ist, werden zunächst die per GPG signierten Schlüsselarchive per Mail oder Mailingliste verteilt. Wenn nun dieses Archiv vorliegt (hier wird angenommen, dass dieses in

/opt/keycache liegt), kann eine Ladestation folgendes Verfahren anwenden, um deren Authentizität zu prüfen:

1. Auslesen von  $I_R$  aus der RFID-Karte
2. Auslesen des Signaturblocks der RFID-Karte.
3. Dekompression des Signaturblocks der RFID-Karte (gunzip) und Prüfen, ob die Signatur gültig ist. Zusätzliche Prüfung, ob das signierte  $I_R$ -Datum gleich der tatsächlich empfangenen  $I_R$  der vorgehaltenen RFID-Karte ist. Nur wenn beide Prüfungen erfolgreich sind, kann die Karte als gültig akzeptiert werden.

Die Dekompression eines roh von der Karte gelesenen Blocks, der z.B. in `signature.data.gz` liegt, ist trivial:

```
$ gunzip signature.data.gz
```

Nach Dekompression existiert eine Datei `signature.data`, deren digitale Signatur anhand des Archivs der öffentlichen Schlüssel aller Regionen geprüft werden kann:

```
$ openssl smime -verify -in signature.data  
-CApath /opt/keycache
```

Der Rückgabewert ( $\$?$ ) gibt hierbei an, ob die Überprüfung der Signatur erfolgreich war ( $= 0$ ) oder nicht ( $\neq 0$ ).

## 4 Sicherheitsanalyse

Es soll kurz erläutert werden, welche Angriffe durch oben beschriebenes Verfahren abgewehrt werden können und gegen welche es anfällig ist.

### 4.1 Fälschen von Karten

Das Fälschen von Karten, also das Erzeugen falscher digitaler Signaturen ist nach heutigem Stand der Technik bei den beschriebenen Verfahren praktisch unmöglich. Auch das Erzeugen eines eigenen Schlüsselpaars und aufbringen einer

eigenen Signatur wäre für einen Angreifer nicht zielführend, da die Signatur einer Überprüfung nicht standhalten würde, da nur öffentliche Schlüssel von vertrauten Einheiten in den öffentlichen Keycache gelangen können.

Der Keycache muss hierfür selbstverständlich entsprechend gegen unautorisierte Veränderung geschützt sein: Sobald es einem Angreifer nämlich gelingt, in den Keycache eigene Zertifikate einzubringen, wird das System möglicherweise auch unautorisierten Karten (d.h. Karten, die mit  $K_{\text{Priv}}$  des Angreifers signiert sind) den Zugriff gewähren.

## 4.2 Kopie von Karten

Ein Kopieren von Karten (also Auslesen einer gültigen Karte und Aufspielen der Daten auf eine andere Karte) ist nicht möglich, da dadurch die Signatur wegen der anderen RFID-ID der neuen Karte ungültig wird.

## 4.3 Emulations-Angriff

Wenn  $K_S$  einem Angreifer bekannt ist, kann dieser eine gültige Karte auslesen und mit Hilfe eines speziellen technischen Geräts emulieren. Ein solches Gerät wird im Folgenden *RFID-Emulator* genannt. Beispiele für derartige Schaltungen sind der Proxmark3 oder das HF RFID Demo Tag der TU Graz. Da nicht davon ausgegangen werden kann, dass  $K_S$  beim Teilen durch mehrere Ladesäulenhersteller dauerhaft geheim bleibt, ist die Emulation ein vorstellbares Angriffsszenario. Weder der Proxmark3 noch das HF RFID Demo Tag ist allerdings durch die momentan erhältliche Firmware im Auslieferungszustand dazu in der Lage, eine DESFire EV1 Karte derart zu emulieren. Beide Hardware-Plattformen bieten nur jeweils eine einfache Abstraktion, den darunterliegenden ISO-14443A-Layer anzusteuern. Deshalb ist erhebliches technisches Verständnis und Ressourcen erforderlich, um einen solchen Angriff durchzuführen, da diese Funktionalität selbst in Software nachgerüstet werden müsste. Ein solcher Angriff scheint auch nicht sonderlich prestigeträchtig, da bei symmetrischen Kryptosystemen hinlänglich bekannt ist, dass diese nicht mehr sicher sein können, falls  $K_S$  kompromittiert wird. Vom Standpunkt einer finanziellen Bedrohung ist dieser Angriff auszuschließen, da es ungleich einfacher ist, sich einer gültigen Karte durch Diebstahl zu ermächtigen.

## 5 Wahl des symmetrischen Schlüssels

In jedem Fall ist bei Verwendung von RFID-Karten immer ein symmetrischer Schlüssel  $K_S$  zu verwenden – prinzipbedingt gibt es nicht die Möglichkeit, keinen Schlüssel zu verwenden. Stattdessen gibt es aber selbstverständlich die Möglichkeit,

einen Nullschlüssel (WKK) zu verwenden. Welche Variante letztendlich zum Einsatz kommt, soll im folgenden Abschnitt erläutert werden.

## 5.1 Alternativen

Das grundlegende Problem bei Verwendung eines symmetrischen RFID-Systems ist, dass bei Bekanntwerden des symmetrischen Schlüssels  $K_S$  ein Emulationsangriff, wie in 4.3 beschrieben, möglich wird. Eine Verwendung von Mifare Classic statt Mifare DESFire hat exakt dieselben inhärenten Probleme eines symmetrischen Kryptosystems, hat allerdings zusätzlich die gravierende Sicherheitslücke eines gebrochenen Verschlüsselungsalgorithmus. Bei DESFire kommt die nach heutigen Standards als sicher erachtete Cipher AES zum Einsatz, bei Classic ist dies Crypto1. [8] Crypto1 gilt seit 2006 als gebrochen, seit 2008 gibt es hochgradig effiziente Angriffsmöglichkeiten, die die Sicherheit des 48-Bit Crypto1-Schlüssel quasi auf Null reduzieren. Die Sicherheit bei Mifare Classic ist also – im Gegensatz zu Mifare DESFire – unabhängig vom Schlüssel, wie auch in Abbildung 2 zu sehen ist.

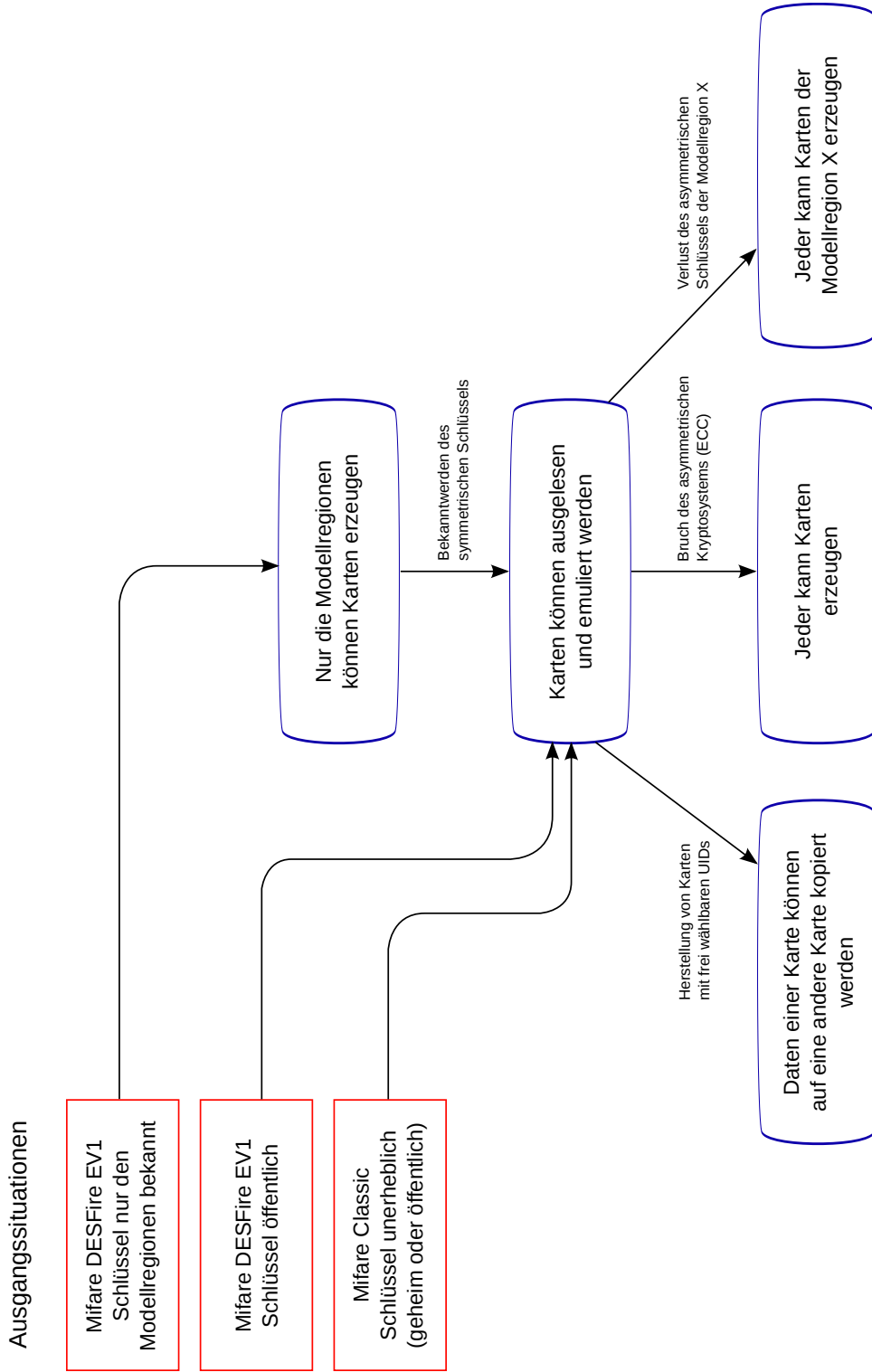


Abbildung 2: Verschiedene Ausgangsszenarien und deren Implikationen auf die Systemsicherheit

## 5.2 Plausibilität des Angriffsszenarios

Nachdem die Angriffsszenarien erneut erläutert wurden, muss hinterfragt werden, wie plausibel tatsächlich ein Emulations-Angriff in der Praxis ist. Zusammenfassend die Schritte, die für eine Kompromittierung des Systems notwendig wären:

1. Bekanntwerden des symmetrischen Schlüssels  $K_S$  durch Verlust durch einen Ladesäulenhersteller.
2. Beschaffung eines RFID-Emulators durch einen Angreifer. Modifikation der Firmware des Emulators, um entsprechenden Angriff durchzuführen.
3. Auslesen einer gültigen RFID-Karte und abspeichern der darauf aufgebracht- en Informationen.
4. Konfiguration des RFID-Emulators, um die ausgelesene Karte inklusive der ausgelesenen Informationen in ein RFID-System wiedereinzuspielen.
5. Benutzung des Emulators, um einer Ladesäule vorzutäuschen, dass sich die vorher ausgelesene Karte im Nahfeld des RFID-Kartenlesers befindet.

Zunächst ist hier also notwendig, dass einer der Teilnehmer den Schlüssel versehentlich bekanntmacht. Es genügt hierfür, dass ein einziger Teilnehmer den Schlüssel verliert, da die Schlüssel modellregionenübergreifend identisch wären. Sollte dies passiert sein, muss ein Angreifer, der über Expertenwissen im Bereich RFID verfügt, Kenntnis dieses Schlüssels erhalten. Damit kann er Spezialhard- ware so programmieren, dass sie eine ausgelesene Karte emulieren kann. Hierzu muss der Angreifer natürlich auch – zumindest für kurze Zeit – physikalischen Zugriff auf eine gültige Karte haben. Danach kann an einer Ladesäule diese Karte emuliert werden, also mit Hilfe der Spezialhardware vorgetäuscht werden, dass der Angreifer im Besitz der vorher ausgelesenen RFID-Karte ist.

Ein solcher Angriff ist durchaus im Bereich des Möglichen. Allerdings muss kritisch die Lukrativität eines solchen Szenarios hinterfragt werden. Aufgrund des großen Missverhältnisses zwischen Aufwand und Gewinn ist ein monetär motivierter Angriff praktisch ausgeschlossen. Es bleibt also das – selbstverständlich nicht zu unterschätzende – rein akademisch motivierte Angriffsszenario. Der Prestigegewinn eines in oben beschriebener Art durchgeführten Angriffs ist jedoch hochgradig fragwürdig: Denn dass sich bei Verlust eines symmetrischen Schlüssels in einem Kryptosystem bestimmte Sicherheitslücken auftun ist keine Neuheit, sondern ganz im Gegenteil völlig trivial und allgemein bekannt. Praktisch vergleichbar ist dies mit einem echten Schlüssel einer Schließanlage: Sollte auch nur ein Einziger der identischen Schlüssel verloren gehen, ist es trivial möglich, sich zu den kompromittierten Schlössern Zugriff zu verschaffen. Dies stellt aber

keine akademische Meisterleistung dar, sondern ist ganz im Gegenteil ein eher primitiver, plumper Angriff mit niedrigem Prestigegewinn für den Angreifer – vergleichbar mit dem eines simplen Diebstahls einer RFID-Karte. Viel interessanter sind Angriffe auf Schlösser, die ohne Besitz beziehungsweise Kenntnis des Schlüssels durchführbar sind – solche Angriffe sind nach momentanem Stand der Technik durch die gewählten Verfahren bei korrekter Implementierung allerdings hochgradig unwahrscheinlich.

### **5.3 Empfehlung**

Um Interoperabilität mit Hilfe der aktuell verwendeten Hardware zu erzielen liegt die Idee Nahe, einen symmetrischen Schlüssel zufällig zu wählen und diesen unter allen teilnehmenden Ladesäulenherstellern zu verteilen. Der Schlüssel sollte nach dem Need-to-know-Prinzip an nur diejenigen Personen verteilt werden, die für die Implementierung des Systems tatsächliche Kenntnis des Geheimnisses haben müssen. Je geringer die Anzahl der Kenntnisträger ist, umso höher ist die Wahrscheinlichkeit, dass das Geheimnis tatsächlich über einen längeren Zeitraum geheim bleibt.

## **6 Ausblick**

### **6.1 Notwendigkeit einer Migration**

Die vorgestellte Lösung ist, wie in 3.6 und 4.3 beschrieben, aufgrund der inhärenten technischen Beschränkungen eines symmetrischen Kryptosystems nicht unproblematisch. Zum momentanen Zeitpunkt gibt es allerdings keine industriereife Alternative, die ein asymmetrisches Kryptosystem in Hardware implementiert. Das hier beschriebene Verfahren ist jedoch derart konzipiert, dass bei Verfügbarkeit einer Alternative problemlos auf eine neuere Technologie gewechselt werden kann.

### **6.2 Migrationsstrategie**

Ein RFID-Token, das sich asymmetrischer Kryptografie bedient, wird mit hoher Wahrscheinlichkeit das ECC-Verfahren benutzen, da dies für ressourcenbeschränkte Geräte momentan die geeignetste Wahl ist. Für ECC auf RFID-Tokens gibt es bereits einige Belege der Machbarkeit, allerdings noch nicht in produktreifer Qualität. [15][24][4] Sollten solche RFID-Tokens verfügbar werden, ist es nicht mehr notwendig, eine auf einem Host-Computer erzeugte Signatur auf einen Datenblock aufzubringen – stattdessen müsste lediglich  $I_C$  auf der Karte aufgebracht werden und der auf der RFID-Karte erzeugte asymmetrische öffentliche Schlüssel

durch den asymmetrischen privaten Schlüssel des Ladesäulenherstellers signiert werden. Dies würde den Aufbau eines sicheren Kommunikationskanals zwischen RFID-Leser und RFID-Karte ermöglichen, ohne dass es ein privates, unter allen Teilnehmern geteiltes Datum (wie beispielsweise  $K_S$ ) geben muss und so effektiv den in 4.3 beschriebenen Angriff verhindern.

## Abkürzungsverzeichnis

- **AES:** Advanced Encryption Standard. Synonym für den symmetrischen Verschlüsselungsalgorithmus Rijndael, der 2000 zum Advanced Encryption Standard erhoben wurde.
- **AMK:** Application Master Key, der symmetrische Hauptschlüssel einer DESFire EV1 Application. Durch Authentifizierung mit dem AMK wird die Administration der Application (also z.B. das Anlegen und Löschen von Files) für die Sitzung freigeschaltet.
- **CA:** Certificate Authority. Die Stelle, die Zertifikate ausstellt (also CSRs unterschreibt).
- **CMK:** Card Master Key, der symmetrische Hauptschlüssel für eine RFID-Karte. Durch Authentifizierung mit dem CMK wird bei DESFire EV1 in der gewählten Konfiguration die Administration der Karte (also z.B. das Anlegen und Löschen von Applications) für die Sitzung freigeschaltet.
- **CSR:** Certificate Signing Request. Ein fast vollständiges Zertifikat, dem allerdings noch die digitale Signatur von einer CA fehlt. Durch die Unterschrift wird ein CSR zum Zertifikat.
- **ECC:** Elliptic Curve Cryptography. An elliptischen Kurven existiert eine bestimmte schwer lösbare Problemklasse, auf der ein asymmetrisches Kryptosystem basiert (ähnlich wie RSA, das allerdings auf dem Problem der nichttrivialen Faktorisierbarkeit zusammengesetzter Ganzzahlen basiert).
- **FIPS:** Federal Information Processing Standard. Bezeichnung für öffentliche Standards der USA.
- **GPG:** GNU Privacy Guard. Ein Programm zur einfachen dezentralen Verschlüsselung und Signatur von Daten, hauptsächlich im E-Mail-Verkehr gebräuchlich. Offene Variante von PGP (Pretty Good Privacy).
- $I_C$ : Contract-Identifizier, welcher für jede zu identifizierende Entität (z.B. Kunden) eindeutig ist. Format noch zu definieren.
- $I_R$ : RFID-Identifizier, welcher für jede RFID-Karte eindeutig ist. Für Mifare DESFire EV1 7 Byte lang.
- $K_M$ : RFID-Masterschlüssel, von dem für eine konkrete Karte ein Kartenschlüssel abgeleitet werden kann, siehe auch 2.3.

- **$K_S$** : Symmetrischer RFID-Kartenschlüssel, der für die Kommunikation mit der Karte benutzt werden muss.
- **NIST**: National Institute of Standards and Technology. Eine Bundesbehörde der USA, die für Standardisierungsprozesse zuständig ist.
- **PICC**: Proximity Integrated Circuit Card, offizielle Abkürzung für eine RFID-Karte
- **PKI**: Public Key Infrastructure. Bezeichnet ein üblicherweise hierarchisch aufgebautes asymmetrisches Kryptosystem, in dem bestimmte Autoritäten (CAs) anderen Entitäten vertrauen attestieren können, indem sie deren öffentliche Kryptoschlüssel digital signieren.
- **RFC**: Request for Comments. Ein Dokumententyp, in dem offene Internetstandards (z.B. IPv4, IPv6, TCP, UDP, SMTP, POP, IMAP, HTTP) beschrieben sind.
- **SAM**: Security Access Module. Eine physikalisch wie eine SIM-Karte beschaffene kryptografische Hardwareeinheit zur sicheren Verwahrung symmetrischer Schlüssel für RFID-Systeme. Proprietäres System von Mifare.
- **S/MIME**: Secure Multipurpose Internet Mail Extensions. Ein Containerformat für Daten, welches Signatur und Verschlüsselung unterstützt und im Mailverkehr gebräuchlich ist.
- **WEP**: Wired Equivalent Privacy. Ein von der IEEE verabschiedeter Standard zur Verschlüsselung von Drahtlosnetzwerken. Basiert auf einer gebrochenen Cipher, wodurch Kommunikation in WEP-verschlüsselten Netzwerken mit minimalem Aufwand abgehört oder manipuliert werden kann.
- **WKK**: Well-known Key. Ein Schlüssel, der jedermann bekannt ist und üblicherweise dort verwendet wird, wo kein Schlüssel notwendig ist, das verwendete Verfahren jedoch nicht die Möglichkeit bietet, keinen Schlüssel zu verwenden.

## Literatur

- [1] Arnaud Tisserand. Introduction to Elliptic Curve Cryptography. CNRS, IRISA laboratory, CAIRN research team, 2009. <http://www.irisa.fr/prive/Arnaud.Tisserand/docs/slides-semcairn09-ecc-4p.pdf>.
- [2] BSI. Technische Richtlinie TR-02102, Kryptographische Verfahren: Empfehlungen und Schlüssellängen. BSI, 2008. [https://www.bsi.bund.de/cae/servlet/contentblob/477256/publicationFile/30924/BSI-TR-02102\\_V1\\_0\\_pdf.pdf](https://www.bsi.bund.de/cae/servlet/contentblob/477256/publicationFile/30924/BSI-TR-02102_V1_0_pdf.pdf).
- [3] cv cryptovision GmbH. ECC - Kryptographie auf Basis elliptischer Kurven - Eine kurze Einführung. cv cryptovision GmbH, 2009. [http://www.cryptovision.com/fileadmin/media/documents/Whitepaper\\_Produnkte/02-Whitepaper-Technical-ECC\\_EN.pdf](http://www.cryptovision.com/fileadmin/media/documents/Whitepaper_Produnkte/02-Whitepaper-Technical-ECC_EN.pdf).
- [4] Daniel Hein, Johannes Wolkerstorfer, Norbert Felber. ECC is Ready for RFID – A Proof in Silicon. Faculty of Computer Science Graz University of Technology and ETH Zürich Integrated Systems Laboratory, 2008. <http://events.iaik.tugraz.at/RFIDSec08/Papers/Publication/03%20-%20Hein%20-%20ECC%20is%20Ready%20-%20Slides.pdf>.
- [5] Elisabeth Oswald. Einsatz und Bedeutung Elliptischer Kurven für die elektronische Signatur. A-SIT – Secure Information Technology Center Austria, 2001. [http://www.ecc-brainpool.org/ElliptischeKurven\\_und\\_Signatur\\_Studie.pdf](http://www.ecc-brainpool.org/ElliptischeKurven_und_Signatur_Studie.pdf).
- [6] Erik Tews, Ralf-Philipp Weinmann, Andrei Pyshkin. Breaking 104 bit WEP in less than 60 seconds. TU Darmstadt, 2007. <http://eprint.iacr.org/2007/120.pdf>.
- [7] Federal Information Processing Standards Publication. Digital Signature Standard (DSS). National Institute of Standards and Technology, 2009. [http://csrc.nist.gov/publications/fips/fips186-3/fips\\_186-3.pdf](http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf).
- [8] Henryk Plötz. Mifare Classic – Eine Analyse der Implementierung. Institut für Informatik, Humboldt-Universität Berlin, 2008. [http://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2008-21/SAR-PR-2008-21\\_.pdf](http://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2008-21/SAR-PR-2008-21_.pdf).
- [9] J. Bauer. Beispielschlüsselliste der RFID-Umgebung zur interoperablen Authentifizierung. IKT FG IOP, January 2011. <https://extranet.pt-dlr.de/e-energy/FGInteroperabilitaet-EM/InterneDokumente/TaskForce%20AIEV%20AK%20RFID/KeyMaterialExample.pdf>.

- [10] J. Bauer. Schlüsselliste der RFID-Umgebung zur interoperablen Authentifizierung. IKT FG IOP, January 2011.
- [11] J. Bauer, M. Dipper. RFID Authentication Encryption Alternatives. Innovations Software Technology Bosch Group, 2010. <https://extranet.pt-dlr.de/e-energy/FGInteroperabilitaet-EM/InterneDokumente/TaskForce%20AIEV%20AK%20RFID/2010-06-07%20RFID%20Authentification%20Encryption%20Alternatives.ppt>.
- [12] JH. Song, R. Poovendran, J. Lee, T. Iwata. RFC4493: The AES-CMAC Algorithm. Internet Engineering Task Force, June 2006. <http://www.ietf.org/rfc/rfc4493.txt>.
- [13] Jie Liang, Xuejia Lai. Improved Collision Attack on Hash Function MD5. Department of Computer Science and Engineering, Shanghai Jiao Tong University, 2005. <http://eprint.iacr.org/2005/425.pdf>.
- [14] Joachim Swoboda, Stephan Spitz, Michael Pramateftakis. *Kryptographie und IT-Sicherheit: Grundlagen und Anwendungen*. 2008.
- [15] Johannes Wolkerstorfer. Is Elliptic-Curve Cryptography Suitable to Secure RFID Tags? Institute for Applied Information Processing and Communications, Faculty of Computer Science Graz University of Technology, 2005. <http://events.iaik.tugraz.at/RFIDandLightweightCrypto05/RFID-SlidesandProceedings/Wolkerstorfer-ECC%20and%20RFID.pdf>.
- [16] Jovan Dj. Golić. Linear Statistical Weakness of Alleged RC4 Keystream Generator. School of Electrical Engineering, University of Belgrade, 1999. <http://www.wisdom.weizmann.ac.il/~itsik/RC4/Papers/Golic.PDF>.
- [17] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. NIST, May 2005. <http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf>.
- [18] NXP. MF3ICD81 - MIFARE DESFire Functional Specification, Rev 3.5 (Doc 134035). NXP, November 2008.
- [19] NXP. AN094533 - MIFARE DESFire EV1 - Features and Hints, Rev 3.3 (Doc 094533). NXP, December 2009.
- [20] NXP. MF3ICD21, MF3ICD41, MF3ICD81 - MIFARE DESFire EV1 contactless multi-application IC. NXP, March 2009. [www.nxp.com/documents/short\\_data\\_sheet/MF3ICD21\\_41\\_81\\_SDS.pdf](http://www.nxp.com/documents/short_data_sheet/MF3ICD21_41_81_SDS.pdf).

- [21] NXP. AN10922 - Symmetric Key Diversification, Rev 1.3. NXP, March 2010. [http://www.nxp.com/documents/application\\_note/AN10922.pdf](http://www.nxp.com/documents/application_note/AN10922.pdf).
- [22] P. Deutsch. RFC1952: GZIP file format specification version 4.3. Internet Engineering Task Force, 1996. <http://tools.ietf.org/rfc/rfc1952.txt>.
- [23] S. Turner, D. Brown, K. Yiu, R. Housley, T. Polk. RFC5480: Elliptic Curve Cryptography Subject Public Key Information. Internet Engineering Task Force, 2009. <http://tools.ietf.org/html/rfc5480>.
- [24] Sandeep Kumar, Christof Paar. Are standards compliant Elliptic Curve Cryptosystems feasible on RFID? Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, 2006. <http://events.iaik.tugraz.at/RFIDSec06/Program/slides/006%20-%20Standards%20Compliant%20ECC.ppt>.

## Stichwortverzeichnis

$I_C$ , 8

$I_R$ , 6

$K_M$ , 7

$K_S$ , 6

$K_{Priv}$ , 6

$K_{Pub}$ , 6

AES-Key, 6

Application, 7

Contract-ID, 8

File, 7

Keyblob, 12

Master-Key, 7

RFID-Emulator, 18

RFID-ID, 6

## Appendix A: Testvektoren

Die Testvektordateien sind hier in textueller Repräsentation im Base64-Encoding gegeben und ihre jeweilige SHA1-Prüfsumme ist zusätzlich ersichtlich. Dies macht es möglich, die Dateien auch ohne separaten Anhang mit minimalem Aufwand bytegenau aus dem Dokument herauszukopieren. Eine Testdatei, die lediglich den Inhalt „FooBar“ (ohne CR oder LF am Zeilenende) enthält, sieht also kodiert so aus:

```
Rm9vQmFy
```

Datei 1: FooBar.txt: 6 Bytes, eb8fc41f9d9ae5855c4d801355075e4ccfb22808

Um diese Datei zu dekodieren kopiert man den obigen Textblock in eine Datei (z.B. FooBar.base64) und führt folgenden Befehl aus:

```
$ openssl base64 -d -in FooBar.base64 -out FooBar.txt
```

Als Testvektoren sind zwei Zertifikate von zwei imaginären Ladesäulenherstellern gegeben, der „Modellregion Bodensee“ und der „Modellregion Umlautingen“. Für erstere sind zwei Keyblobs angegeben: Der erste ist für die  $I_R$  deadbeefc0ffee mit der  $I_C$  „DE-BO-123456“ und enthält eine gültige Signatur. Der zweite gibt vor für die  $I_R$  ababababababab mit der  $I_C$  „DE-BO-999999“ zu sein, enthält aber eine ungültige Signatur. Im Zertifikat für die Modellregion Umlautingen können Umlaute getestet werden (diese sollte im OU den String „Test AÄOÖUÜäöüüß“ enthalten). Auch für diese Region ist ein gültiger Keyblob gegeben mit  $I_R$  d7894a066cb8e3 und  $I_C$  „DE-UM-31415“.

LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUJ2akNDQVcwQ0FqQTVN  
QWtHQnlxR1NNNDlCQUV3ZFRFTE1Ba0dBMVVVFQmhNQ1JFVXhDekFKQmdOVkJB  
Z1QKQWtKWE1STXdFUVlEVlFRSEV3cEpiVzFsYm50MF1XRmtNU3N3S1FZRFZR  
UURFeUpOWlZKbFoybHZUVzlpYVd3ZwpUVzlrWld4c2NtVm5hVz1lSUVKdlpH  
VnVjMlZsTVJjd0ZRURWUVFMRXc1S2IyaGhibTVsY3lCQ1lYVmxjakFlCkZ3  
MHhNREEzTURZeU1EQTFNekZhRncweU1EQTNRE15TURBMU16RmFNSFV4Q3pB  
SkJnTlZCQVlUQWtSRk1Rc3cKQ1FZRFZRUU1Fd0pDVnpFVE1CRUdBMVVFQnhN  
S1NXMXRaVzV6ZEdGaFpERXJN2tHQTFVRUF4TWlUV1ZTWldkcApiMDF2WW1s  
c0lFMXZaR1ZzYkhKbFoybHZiaUJdYjJSbGJuTmxaVEVYTUJVR0ExVUVDdeE1P  
U205b1lXNXVaWE1nClFtRjFaWE13VGpBUUJnY3Foa2pPUFFJQkFnVXJnUVFB  
SVFNNkFBVEU4U24vZms5c0Q1TjVXaGt1NExmZDR0QXEKVnNBWng3QTA5RGRr  
SXVwejI2SkFZQytBYlpva3ZVSTRlS3N2MDRTaUJST0VlMU1JdlRBSkFnY3Fo  
a2pPUFFRQgpBMEFBTUQwQ0hRQ3RZdzB6ZTY1a0VmM054aHV2U2NDeVZXaURh  
bzFQTTBRM21RaDFBaHcyYTIzVDYwNzd3Mn1QCm4rNHV1K3kwdHJQUHVjbDdr  
TF15bGZndAotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==

Datei 2: Bodensee.crt: 664 Bytes, dbf7422115701be62873b2f4836358d30b2b73fe

LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUIwakNDQVlBQ0FRRXdD  
UVlIS29aSXpqMEVBEIvTVFzd0NRWURWUVFHRXdKRVJURUxNQWtHQTFVRUNC  
TUMKUWxjeEZEQVNCZ05WQkFjVEMxVnRiR0YxZEEdsdVoyVnVNUzR3TEFZRFZR  
UURFeVZOWlZKbFoybHZUVzlpYVd3ZwpUVzlrWld4c2NtVm5hVz11SUZwdGJH  
RjFkR2x1WjJWdU1SMHdHd11EVlFRTEZCUlVaWE4wSUVIRVQ5W1YzR0hrCmIv  
WjEveSE56M3pBZUZ3MHhNREEzTURZeU1EQTFNekZhRncweU1EQTNRE15TURB  
MU16RmFNSDh4Q3pBSk1JnTlYkQkFZVEFrUkZNUXN3Q1FZRFZRUU1Fd0pDVnpF  
VU1CSUdBMVVFQnhNTFZXMjZWFYwYVc1blpXNHhMakFzQmdOVgpCQU1USlUx  
bFVtVm5hVz10YjJKcGJDQk5iMlJsYkd4eVpXZHBiMjRnVlcxc11YVjBhVzVu  
Wlc0eEhUQWJCZ05WckJBc1VGRlJsYzNRZ1FjU1AxbFhjWWVsdjluWDh1M1Bm  
TUU0d0VBWUhlL1pJemowQ0FRWUZLNEVFQUNFRE9nQUUKUmRXeTAwRGtOZ09V  
U1Z5eHpiMjJmSGNkMEZ6V245aUtOUXRvWnFSTUREbDd0TWJ6UVNkdDR2cHQy  
QjhYS01Nawo2Qmk4RnVBVHJQSXdDUVlIS29aSXpqMEVBUU5CQURBK0FoMEE1  
RVhtcXVQc1p5bndYS112M1dkdzVSUEFOTzNjCk1TckdwSUNtckFJZEFOcWI1  
U2ZTSVE3WEJZa1NuV3czOVVKeW04aDhGYkFhbXRxT212Yz0KLS0tLS1FTkQg  
Q0VSVElGSUNBVEUtLS0tLQo=

Datei 3: Umlaut.crt: 692 Bytes, 65aa9d8f57dc0f6d2ab570600ffa1f7ad68d62d2

H4sIAIuMM0wAA41UWY+iTBR951eYfu047YLaTqcfqooCUYsW2dQ3QCx2ENBC  
fv1XttOzZJIvQ0huhHWXc889BVEJ7ttBVUdF/r03/DYQUJE3Qd70zVsZf091  
17SJSrdqXuqI5sHxrVdWRVP4Rfr+5JZlGvluw1Nf2n6Z+PWsf49ym0sVPL31  
Mn6Y0vc6dIdvPa+45Ee3ur0/9fkDZ+OJLIsihFgEkwGSR+IQyBIYIDydjRF+  
EgQzjOoef928Z7wQDrP3QNDLgrp2aSAI/f6/lBIk3Icf/eFoLE6m/WPgHr0g  
OPmD0ykI/rXEN5z879xvvdzNgvenOouy4Fs5q59+ZVduXp+Cqo9zvzhGOf3e  
89w6mIo/I6SoLos6aj6X4Tan64cZ97/1TlEa/FVXIKoqxWC/XBUHNbz6GtB9  
VHDfyUQIjBMEDNyiDiwhtSqJEqRQCxBQK8g4K4bqjQVJxwvvg8mBfZZQqz7wg  
9HQVESlHsN8v7oW7eICBCYdEr+8+ydZ1BbMl3pp4LRCQKGB0YQQJ0l0/xR3Y  
QqrZEPgmStLgc+yLP96Ge0dvVx0oH2fEVJNhamV2LrjOXPNgy9JDkNtt6int  
7eAcS28kUj2bJwdn0h0cq5U7YD+Sa1NKyquryBcvsztbxcvw6Ng3AsWdZGJG  
JJ9pscqt1RG74D7100e6nz525NC/kAswJGiLrR9E3TtQEYTLHdkShh/zLjAr  
154zTL1cG+wdOSFGzVaPM0nAt6V2sJfpYZReTWceUQ6j3HLobe3zGfmIF47y  
elA4FyM7JVufyY/kNwaTleCNwtDLJql/g2i/s1M/hh8EAgXe+JqIOAcUK9C4  
QAhaCLoCQGK2q9P82RyxTTJOXUEXzrSZj6MBKh0G80UtmQntPEftfOMVCaRr  
ETZZZMz10BTptxczXKkyFrczi0XzOw/CVycIsKQDwDa6egSrYSyZ2qzK902L  
PlaU6XzJ1kvbVbVzbL12Wyk4oIWFftKhErzjppjKvV81Tn0+zaDqbrY1R/fo8  
ue0ueh0v7Gc0aLnAjLvAYqpCLqgZWfwiXYBgz1nfyr+pTOUqQ3aHTQLx56pg  
S1aGM2zumjgqcnIQcEXQjzWClkSmYxuf4hkMr/ssrVU8vLNee4vHdriok+R  
5Vp6MPGOQ0uh3pZ8CEY2L/b05HLYEa48eXjYqQjEYHInCBrlB2CqC3VEqYeZ  
8selSySOWhG+btXnpTKBD2lyDpNImbMBRLrVSq2vERMwrQMjEnNrktajQfwV  
JzwC9VamujWB6W3nTNRsxFVcyyGiScq7LK9cRwW/ap7hlgW0zuwEic1Xx01  
2IGP378B5eOogEn6Xv2ZzLjPAgyyvbxiQIA6v8dYFCG2G//At0CEHONUK6R  
tcjFNNPPmF5GSXsRXqG+nGTTpIxX2vSm0onXRGa3RwsdVa9eUybtzBxI5400  
voUnrSZZXCQT/bxcqluwiZvwX3/q/b4g/AcVBq2WuwYAAA==

Datei 4: Bodensee-deadbeefc0ffee.sig: 1114 Bytes,  
06e809f1a35741c243888203df8f1c2cae31b4fe

H4sICOCOM0wAA3gAjVRbb+I8EH3Pr0B9rdhyCVBa9cF2TAjgFJMb8JYE6tyT  
JgGH/PrPwHa3q5U+bRJplPHM+MyZYxON4K59LKswz146/R89CeVZfczqrnkp  
ji+d9JTUYeGW9VMVsux4e00UZV7nfp68PbhFkYS+W4vUp6ZbxH416V6j3PpU  
Hh9e061YTNhbFbj9146Xn7KDW17eHrrigZPhaDaTZQixDEY9NBvIfTBTQA/h  
8WSI8IMkmUFYdcTnZh3jiQiYnTuCTnqsKpcdJanb/ZdSkoK78L07vT1d1/v+  
/muJPzn5375f05mbHt8eqjRMjz+KSfXw07t0s+rjWHZx5ueHMGmVhc+tjmP5  
V4QSVkVehfVtGG5du36QCv9r5yNMjn/VlYimKRHYLZb5XgvOvg6oj3Lh+zAR  
AsMYAYoblIIFZFapMIJUzgECKhUZn6qheUNJoXgOXBHsa5wx9VEUhB7VEFFi  
juhufi3cRj0MTNngtLr6FJtSFfMF3ph4JREQq6BvYQQJoonf4BZsINNtCHwT  
xUntOfbJH26CnUObZQuK+xoxtbifWKmdSa4z1b3BovAQFHaTeGpz2TuHwhvI  
jKbTeO+M2r1jNbMW2PfkylTi4uyqs5OX2q2k4UVwcOwLgfJWMTEnis/1SBPW  
aomdC59285H2148fBPQv5BIMCNpg6ydr1x2YCeLFlmwIx/d+55gXC8/pJ16m  
93bOLCZGxZf3NUXC14W+txfJfpCctWcaug5nwgroTeWLHkWLJ4HyvFcFFwM7  
IRufz+7JK8xHS8kbBIGXjhL/AtFuayd+BN8JBCq8iDEReQoYVqFxxghCAALQ5  
gMRslh/TR3PA1/EwcSXVmlB6Ogx7qHA4zOaVYvb0AQ3H2drLY8hWMqzT0JJS  
KJA/L09msNRmWN5MLB5OrzxIXztBgBUKAF9T7QCW/Ugx9UmZ0s2TP1TV8XTB  
Vwvb1fTPyHpuN8pxj+YWmiv7UvIO69I8nxVPe/yYhOPJZGUMqufH0WV7onW0  
tB9RrxECM64Ci5gGhaAmZP6bdAmCnWB9M/umMk2oDNktNgnEt1HBhiwNp19f  
NXFQZ8FewSVBP8cIGhKajm3cxNPrn3dpUmm4f2W98ub36XghRDeRZXqyN/GW  
QOuu3oa8S0Y6zXf06LTfEqG8WX+/1RCIwOhKEDSqd8A1F1LEmIe5+sehixWB  
WpW+TtXtUJnAhyz+DOJQnfIeRNRqlMbXiQm43oIBiYQ1SeNEIPqKk+6BtJkx  
aolgctk6I23waITlFMtB7MkqP20uAofF/th9TSyL0xYsJcjY144KbMH793/A  
RDsa4Ardab+SufBZgEO+my25rEiA+X8HGIwj/u0uoBaAWGiECY2sZCGmCf3E  
7DSIm5P0D0lilI7jI1rq44vGRl4dmu0OzSkqn726iJuJ2VM+18rwEnzoFUmj  
PB7Rz8VK24A1WQf/eq13u5L0H++165u7BgAA

Datei 5: Bodensee-invalid.sig: 1107 Bytes,  
7a186c41143d80ec90e7b4417d2f035a93e5abb8

H4sIAIuMM0wAA42UXXOqOhSG7/kVTm/3sBVBhXZ6kQAiarRBQPGOLyPKlwYN  
5ddvtGe3u6dnnynDTCZv1npreRNkI103o3PNCnyx47ws8epRV7FecXbr2X8  
2MkuaZWU/rnq0oTkcfTUKc9FVYRF+vzgl2WahH7VpnZrvjyGdMTfovzqco4f  
njpZu5iSZ7r3hadOUFzyyD+/Pj/w7TeWBR3Ko4Ew7I8GY0Uby/poCIYjUZVE  
RRGlB46z9wnttL+fd1Zd1JbZeaug8WU+iTmOJ7/DorTdN5BvChIwoCPRrIi  
+b3hMAzkWPwu4fOW/G/bT53cz+LnB5olWfyzHNGHj+yzn9NdfOb1PCyiJCeP  
ncCn8fCDryW0LGhS3c/Cryo/3Get/tTZJWn8hcsh09ROxJvOiq25v4YLgE01  
aLwSUVUwCFWA9VptwBQS56wRpBrEAQhQQ12djJUziJyG9Qnw2+DIY4RMVi0Q  
EmBCYN+SwtP+eFi+YAZRRP4HtHAh8GxwtMYIU8ap2NNcjE2dTVW30R0ETQMI  
jg5rNHfXAvU2bs9fD/LtWqrnB0DfAMieOkLqZG70+Wt1EfSnZaDCdrTSwKhf  
t+uoDPoS+TdgYoPgDUCd8dhKQxETDofWi5BuQi+2rkq+kUPxZYegtNFsnSEt  
ZIUd2Y5Og9yilcy7hpp3jXG7Rp8jcHwre49US3f+6JW0vU43yMJJ/dWJzqr  
3GBd76ONVQZZ1HLBQb8gld0BoEapvXZX9xZ6wtXLUmrqwnVruDSYTNNtP70G  
CfwCiBCkd4BaY8c13CZSIaz17tqN6v3A6Pa jXd2EiszsA8AfJ2PC29FyBGNg  
YjQEwBoLwdzGS9wHA2u6mU8W12ArR7WINwt/1zcLQRWsrGy3Zrmhvfq6gNO+  
mHA/MtGb1GFBXu3CmLO1BJdUPnz2A0jpAGmSOsHa0Yr9k/Sjzmc7GJZJN+K2  
opbqMuiNI1lbPqH8kM5+CfQ/0y2s6Vcy9VoXjxJpFgbYTcd4LXi fMpfDk94vh  
TGkM1QNb1jrQ8ADRITIN/W+O+5vhu0867m+G477ruP8y3K1q7mYiuKJLwEwf  
YpWQQG9N8eftPGptC++37375bBBCcmw30TE4hfWgip1aq8MFsgFbNKCPDu1o  
o3p9AIf fgW9xuB4T7JRBpdudN0v7Epgci11HPu2W0b5K1KvDnE9vw/EFOQ7D  
DZhBQn6DNNiA5e/5rQINEFMFJmAA9sz3ZNZqDmCQeeMZk9qY8Ov6irTvAfMm  
N7E5tG7BD0A6YC/EjMDU9vw5qk7iQv6xBvLs3Jo/0sJ1culrqVTYymkU+VT1  
JlgNCjqL3G2KR0yBUnd8HcfRRaU5TLaxB867K7aGo+8+/jzPcb8AObKok+MG  
AAA=

Datei           6:           Umlaut-d7894a066cb8e3.sig:           1037           Bytes,  
29fe5cb85ebc960308578243d75fa80b802bab24

## Appendix B: Zugriff auf DESFire EV1 Karte

Zur Vorbereitung des RFID-Karten müssen unter anderem Schlüssel gesetzt werden. Diese können dem separaten Dokument [10] entnommen werden. Beispiele, die öffentlich sind, werden sich immer auf die veröffentlichten Schlüssel aus [9] beziehen. Die beiden Dokumente unterscheiden sich lediglich in den Schlüssel, ansonsten sind sie völlig identisch. Zum Testen kann also problemlos das Schlüsselsystem aus [9] benutzt werden – sobald dies funktioniert, können dann die in [10] spezifizierten, richtigen Schlüssel verwendet werden.

Im Auslieferungszustand ist wichtig, dass die DESFire EV1 Karte im Kompatibilitäts-Modus (DESFire Native) ist, also prinzipiell zunächst nur Single DES macht. Eine Authentifikation gegen die Karte ist hier mittels Native TDES oder Standard TDES möglich. Zum Vorbereiten muss allerdings der PICC auf den AES-Modus gehoben werden. Hierfür muss einmalig mit dem Nullschlüssel eine Standard TDES Authentifikation durchgeführt werden (wie z.B. in [19] Abschnitt 6.2.1 beschrieben), danach muss auf einen AES-Schlüssel gewechselt werden (beschrieben z.B. in [19] Abschnitt 9.1.4 „ChangeKey Command for changing default PICC master key to AES key“). Für das System wird als CMK gesetzt der 128-Bit Schlüssel Nummer 1 verwendet, welcher mit der DESFire UID diversifiziert wurde.

Beispielhaft wird hier nun ein Schlüssel Schritt für Schritt diversifiziert, um zu zeigen, wie die Diversifikation genau funktioniert. Als AES-Schlüssel werden die im Beispieldokument angegebenen Schlüssel verwendet. [9] Leerzeichen und Zeilenumbrüche haben keine besondere Bedeutung und dienen lediglich der besseren Lesbarkeit.

CMK nicht diversifiziert (128 Bit, Nummer 1)	b2da388863e031db03c89665ae64ecd7
UID der Karte	04554e71bd2280
Application ID	000000 (PICC)
System Identifier	464720494f50
AES128 Diversifikations-Konstante	01
Diversifikations Input	01 04554e71bd2280 000000 464720494f50
Länge des Diversifikations-Input	17 Bytes
Benötigte Länge Padding	32 Bytes - 17 Bytes = 15 Bytes
Padding-Pattern	80000000000000000000000000000000
CMAC-Input	01 04554e71bd2280 000000 464720494f50 80000000000000000000000000000000
CMAC $K_1$	2be3175a67009d9439da73b1d3a7aa84
CMAC $K_2$	57c62eb4ce013b2873b4e763a74f5508
CMAC Key	$K_2$ (Padding notwendig)
Initialer IV	00000000000000000000000000000000
AES Input	0104554e71bd2280000000464720494f
IV nach AES	0e9220423bb08c29567f017a472915b5
IV nach XOR mit CMAC Key	59540ef6f5b1b70125cbe619e06640bd
AES Input letzter Block	50800000000000000000000000000000
AES Output letzter Block	99c0e8e01186a501d53a7f00f42e3fdb
Diversifizierter CMK	<b>99c0e8e01186a501d53a7f00f42e3fdb</b>

Tabelle 1: Diversifikation eines DESFire EV1-Schlüssels für den CMK (Achtung, Beispielschlüssel!)

AMK nicht diversifiziert (128 Bit, Nummer 2)	dec298cc5a6654e92c610f40899d4a8d
UID der Karte	04554e71bd2280
Application ID	494b54
System Identifier	464720494f50
AES128 Diversifikations-Konstante	01
Diversifikations Input	01 04554e71bd2280 494b54 464720494f50
Länge des Diversifikations-Input	17 Bytes
Benötigte Länge Padding	32 Bytes - 17 Bytes = 15 Bytes
Padding-Pattern	80000000000000000000000000000000
CMAC-Input	01 04554e71bd2280 494b54 464720494f50 80000000000000000000000000000000
CMAC $K_1$	c5fcbf240ccb24bd14dcffcc6ff50d79
CMAC $K_2$	8bf97e481996497a29b9ff98dfea1a75
CMAC Key	$K_2$ (Padding notwendig)
Initialer IV	00000000000000000000000000000000
AES Input	0104554e71bd2280494b54464720494f
IV nach AES	b43a9f2277b588b4e35ab8d9efeb88db
IV nach XOR mit CMAC Key	3fc3e16a6e23c1cecae34741300192ae
AES Input letzter Block	50800000000000000000000000000000
AES Output letzter Block	125dc934743546a3e50e17131dc0b350
Diversifizierter AMK	<b>125dc934743546a3e50e17131dc0b350</b>

Tabelle 2: Diversifikation eines DESFire EV1-Schlüssels für den AMK (Achtung, Beispielschlüssel!)

Filekey nicht diversifiziert (128 Bit, Nummer 3)	eae219cdb7829b8f2d90a721ef1cb156
UID der Karte	04554e71bd2280
Application ID	494b54
System Identifier	464720494f50
AES128 Diversifikations-Konstante	01
Diversifikations Input	01 04554e71bd2280 494b54 464720494f50
Länge des Diversifikations-Input	17 Bytes
Benötigte Länge Padding	32 Bytes - 17 Bytes = 15 Bytes
Padding-Pattern	80000000000000000000000000000000
CMAC-Input	01 04554e71bd2280 494b54 464720494f50 80000000000000000000000000000000
CMAC $K_1$	f18df644a121b19d12ba95054331d87
CMAC $K_2$	e31bec8894243633a25752a0a8663b89
CMAC Key	$K_2$ (Padding notwendig)
Initialer IV	00000000000000000000000000000000
AES Input	0104554e71bd2280494b54464720494f
IV nach AES	1d13da50ca5ba6553d7e6eaf13c3c412
IV nach XOR mit CMAC Key	fe0836d85e7f90669f293c0fba5ff9b
AES Input letzter Block	50800000000000000000000000000000
AES Output letzter Block	e7595d6a8975735e242d6ec2bb1adc64
Diversifizierter Filekey	<b>e7595d6a8975735e242d6ec2bb1adc64</b>

Tabelle 3: Diversifikation eines DESFire EV1-Schlüssels für den lesenden Dateizugriff (Achtung, Beispielschlüssel!)

Danach müssen die DESFire PICC Einstellungen mittels dem ChangeKeySettings Kommando verändert werden: Gültige RFID-Karten haben folgende Konfiguration für die PICC Master Key Settings:

- Configuration changeable: 1
- PICC master key not required for create/delete: 0
- Free directory list access without PICC master key: 0
- Allow changing the PICC master key: 1

Welchen Bits diese Einstellungen entsprechen, kann [18] im Abschnitt 9.3.4 entnommen werden.

Nun befindet sich die DESFire EV1-Karte in einem Zustand, in dem Sie einen diversifizierten CMK benötigt, damit Applications erstellt oder ausgelesen werden können. Nach einer Authentifizierung mit dem diversifizierten CMK kann nun die Application gemäß 3.6 erstellt werden, d.h. es wird eine Application mit AID 0x494b54 erstellt. Hierbei ist zu beachten, dass die AID mit dem LSB zuerst (also Little Endian Encoding) an die Karte gesandt werden muss, d.h. das Byte mit Offset 1 des CreateApplication Kommandos muss 0x54 sein. Für das erstellen der Application werden zwei Konfigurationsbytes gefordert, Key Settings 1 und 2. Das Kommando selbst ist in [18] im Abschnitt 9.4.1 näher beschrieben, die Encodierung der Settings Bytes in 9.3.4 (Key Settings 1) und 9.4.1 (Key Settings 2).

Für das Key Settings 1 Byte (auch als Application Master Key Settings Byte bezeichnet) sind folgenden Einstellungen vorzunehmen:

- ChangeKey access rights: 0 (AMK authentication is necessary to change any key)
- Configuration changeable: 1
- Free create/delete without master key: 0
- Free directory list access without master key: 1
- Allow change master key: 1

In den Key Settings 2 wird codiert, wie viele Schlüssel die Application verwenden wird und von welchem Typ sie sind. Darüber hinaus kann ein ISO/IEC 7816-4 File Identifier Modus aktiviert werden, welcher allerdings im Rahmen dieses RFID-Konzepts nicht benutzt werden wird. Die genauen Einstellungen, die vorzunehmen sind:

- Number of Keys: 2 (wenn kein eigener Schlüssel zur beschleunigten Identifikation verwendet werden soll) oder 3 (wenn ein eigener Schlüssel hinterlegt werden soll, mit dem eigene Karten schneller ausgelesen können werden sollen).
- ISO/IEC 7816-4 file identifiers: 0
- Crypto method of the application: AES

Die genaue Referenz, welche Bits gesetzt werden müssen, sind die Abschnitte 9.4.1 und 9.3.4 von [18].

Nun befindet sich die Karte in einem Modus, in dem ein CMK gesetzt ist und eine Application besteht, die über einen AMK sowie einen oder zwei zusätzliche Schlüssel verfügt. Alle Application-Schlüssel sind allerdings noch WKK (im Auslieferungszustand sind alle Bytes auf 0x00 gesetzt). Diese müssen nun geändert werden.

- Als AMK (Key 0) der Application mit AID 0x494b54 ist der diversifizierte 128-Bit-Schlüssel Nummer 2 zu verwenden.
- Als Key 1 der Application mit AID 0x494b54 ist der diversifizierte 128-Bit-Schlüssel Nummer 3 zu verwenden.
- (optional, empfohlen): Als Key 2 der Application mit AID 0x494b54 ist ein nur im jeweiligen Umfeld bekannter, geheimer Schlüssel zu verwenden, der auch mit der UID der Karte diversifiziert werden sollte, um optimale Sicherheit zu erreichen.

Standard Data Files, die nun auf die Karte aufgebracht werden, müssen folgende Einstellungen verwenden:

- Communication Settings: Fully enciphered communication
- Access Rights: Read Access – Key 1
- Access Rights: Write Access – Key 0 (AMK)
- Access Rights: Read/Write Access – Key 0 (AMK)
- Access Rights: Change Access Rights – Key 0 (AMK)

Die einzige Ausnahme ist das herausgeberspezifische File, welches folgende Einstellungen nutzen sollte:

- Communication Settings: Fully enciphered communication

- Access Rights: Read Access – Key 2
- Access Rights: Write Access – Key 2
- Access Rights: Read/Write Access – Key 2
- Access Rights: Change Access Rights – Key 2

Wie diese Felder genau zu codieren sind, kann [18] in den Abschnitten 8.2, 8.3 und 9.5.5 entnommen werden.