# Abschlussbericht für das Modul SmartCard-Programmeriung

**Implementierung einer Mulitcard-Anwendung**

23. August 2016

# Inhaltsverzeichnis

# 1 Einleitung

In dem Projekt „Implementierung einer MultiCard“ für das Wahlpflichtmodul „SmartCard-Programmierung“ im Sommersemester 2016 geht es um eine beispielhafte Realisierung einer SmartCard-Anwendung. Zu einer SmartCard-Anwendung gehören sogenannte On- und OffCard-Teile. Der OnCard-Teil beschreibt die Anwendung auf der SmartCard, während der OffCard-Teil die Nutzerseite beschreibt.

In dem Projekt „Implementierung einer MultiCard“ sollen mögliche Anwendungen für eine Studenten- und Discokarte realisiert werden. Gleichermaßen soll jegliche Kommunikation kryptographisch verschlüsselt erfolgen, damit eine Manipulation der Karte beziehungsweise der Daten auf der Karte verhindert werden.

**Aufgabe:** Es sollte eine SmartCard-Anwendung entstehen, welche als Bestandteile zum Einen eine Studenten-Karte realisiert. Zum Anderen soll die SmartCard eine sogenannte Disco-Karte realisieren. Weiterhin soll die komplette Kommunikation verschlüsselt erfolgen.

Bestandteile der Student-Anwendung sind:

- Studentenname und Matrikel
- Raumzugänge
- Geld einzahlen und per Essen verbrauchen
- Daten abfragen

Bestandteile der Disco-Anwendung sind:

- Geld einzahlen
- Konsumierte Getränke speichern und bezahlen
- Bonuspunkte für Konsum und verbrauchen der Punkte
- Daten abfragen

Bestandteile der Kryptographie sind:

- Verschlüsseln
- Entschlüsseln
- Signierung

**Verlauf:** Im Verlauf des Projektes wurden die Anforderungen entsprechend der Aufgabe realisiert. Das gesamte Projekt wurde per sogenanntem „Pair-Programming“ realisiert. „Pair-Programming“ bezeichnet das gemeinsame Programmieren an einem Bildschirm, während einer das Programm schreibt denkt der jeweils andere über das geschriebene nach. Regelmäßig werden diese beiden Rollen getauscht.

Aufgrund dass unter Linux programmiert wurde, kann die realisierte Anwendung nicht real getestet werden, da benötigte Treiber nicht existieren. Daher wurde das gesamte Projekt in der Simulation getestet.

**Ergebnis:** Als Ergebnis des Projektes entstand eine kryptographisch gesicherte SmartCard-Anwendung, welche ein Studenten-, Disco- und Crypto-Applet enthält. Außerdem entstand eine OffCard-Anwendung welche diese Funktionalitäten der SmartCard anspricht. Der einfachheitshalber wurde die OffCard-Anwendung in Tabs untergliedern um die verschiedenen Einsatzbereiche abzutrennen.

**Aufbau:** In diesem Bericht wird zunächst auf die verschiedenen Applets mit Ihren Anweisungen und den verschiedenen Fehlern eingegangen. Im Anschluss an den OnCard-Teil wird auf den OffCard-Teil eingegangen. Am Ende des Berichtes wird ein Fazit über das realisierte Projekte gezogen.

# 2 OnCard: Student-Applet

Die AID des Student-Applets ist Student in hexadezimal geschrieben:

- 0x 53 74 75 64 65 6e 74

Das Klassenbyte für das Student-Applet ist:

- 0x 20

Das Student-Applet unterteilt sich in drei Komponenten. Zum Einen die Funktionalität mit Geld umzugehen und zum Anderen das Speichern von Studentennamen sowie die Raumfreischaltungen. Das Essen in der Mensa auf der Studentenkarte wird durch einfaches Geld ausgeben realisiert.

## 2.1 Geld

Im Student-Applet kann Geld eingezahlt und verbraucht werden, dabei wird Euro und Cent jeweils als 1 Byte dargestellt. Auf der Karte wird zwischen ganzzahligem Euro-Betrag und Cent-Betrag unterschieden. Daraus ergibt sich für den Euro-Betrag eine Obergrenze von 127€, da sonst ein Überlauf des Bytes erfolgen würde. Die Höchstgrenze für den Cent-Betrag ist auf 99¢ gesetzt.

**INS_GET_MONEY = 0x20** Liefert das aktuell auf der Karte gespeicherte Guthaben.

**INS_ADD_MONEY = 0x21** Fügt der Karte Geld hinzu.

**INS_SUB_MONEY = 0x22** Es wird ein Betrag vom Geld subtrahiert und damit ausgegeben.

**INS_RESET_MONEY = 0x23** Setzt Euro- und Cent-Betrag auf der Karte auf Null (Werkszustand).

## 2.2 Studentendaten

Auf der Karte wird der Name sowie die Matrikelnummer des Studenten gespeichert. Der Name des Studenten darf maximal 50 Zeichen lang sein.

**INS_SET_NAME = 0x24** Schreibt den Namen des Studenten auf die Karte. Der Name darf maximal 50 Zeichen lang sein.

**INS_GET_NAME = 0x25** Liefert den gepeicherten Namen.

**INS_SET_MATRIKEL = 0x26** Setzt die Matrikelnummer des Studenten (2 Byte)

**INS_GET_MATRIKEL = 0x27** Liefert die gespeicherte Matrikelnummer des Studenten.

### 2.3 Raumfreischaltung

**INS_SET_ROOMS = 0x28** Speichert die als Liste übergebenen Räume auf der Karte.

**INS_GET_ROOMS = 0x29** Liefert eine Liste von gespeicherten Räumen auf der Karte

## 3 OnCard: Disco-Applet

Die AID des Disco-Applets ist Disco in hexadezimal geschrieben:

- 0x 44 69 73 63 6f

Das Klassenbyte für das Disco-Applet ist:

- 0x 30

## 3.1 Bonus

Der Bonus wird als ein Byte gespeichert, daraus ergibt sich der maximale Bonus zu 127, da sonst ein Überlauf auftreten kann.

**INS_GET_BONUS = 0x30** Liefert die aktuell gespeicherten Bonuspunkte.

**INS_ADD_BONUS = 0x31** Fügt Bonuspunkte hinzu.

**INS_SUB_BONUS = 0x32** Verbraucht Bonuspunkte.

**INS_RESET_BONUS = 0x33** Setzt die Bonuspunkte auf der Karte auf Null (Werkszustand).

## 3.2 Geld

Gleiche instructions wie in Studenten-Applet Wird über Applet-Firewall angepsrochen und arbeit mit den Daten des Student-Applets.

**INS_GET_MONEY = 0x20**

**INS_ADD_MONEY = 0x21**

**INS_SUB_MONEY = 0x22**

**INS_RESET_MONEY = 0x23**

## 3.3 Getränke

Maximal Getränke 50.

**INS_GET_DRINKS = 0x34** Liefert eine Liste von konsumierten Getränken.

**INS_ADD_DRINK = 0x35** Fügt ein Getränk der Liste hinzu.

**INS_SET_PAID_DRINKS = 0x36** Anhand einer Übergebenen Liste, welche 1en oder 0en enthält wird die Interne Getränke liste bereinigt um Getränke welche bereits bezahlt wurden. 1 entspricht dabei bereits bezahlt und 0 noch zu bezahlen. Die Übergebene Liste muss exakt die gleiche Anzahl besitzen wie Getränke derzeit gespeichert sind.

Startguthaben mit Konsumbergrenze Bezahlung von konsumierten Getränken beim Verlassen der Disco Bonuspunkte sammeln

# 4 OnCard: Crypto-Applet

Die AID des Crypto-Applets ist Crypto in hexadezimal geschrieben:

- 0x 43 72 79 70 74 6f

Das Klassenbyte für das Crypto-Applet ist:

- 0x 10

**INS_IMPORT_CARD_PRIVATE_MOD = 0x10**

**INS_IMPORT_CARD_PRIVATE_EXP = 0x11**

**INS_IMPORT_CARD_PUBLIC_MOD = 0x12**

**INS_IMPORT_CARD_PUBLIC_EXP = 0x13**

**INS_EXPORT_CARD_PUBLIC_MOD = 0x14**

**INS_EXPORT_CARD_PUBLIC_EXP = 0x15**

**INS_IMPORT_TERMINAL_PUBLIC_MOD = 0x16**

**INS_IMPORT_TERMINAL_PUBLIC_EXP = 0x17** Der komplette Datenaustausch zwischen SmartCard und OffCard-Anwendung soll verschlüsselt und signiert geschehen. Damit nicht jedes Applet die dafür notwendige Logik implementieren muss, wurde mit dem Crypto-Applet eine zentrale Anlaufstelle für folgende Aufgaben geschaffen:

verschlüsseln und signieren

entschlüsseln und verifizieren

Als Kryptosystem wird RSA mit einer Schlüssellänge von 512 Bit eingesetzt. Ursprünglich war eine Schlüssellänge von 1024 Bit angedacht, jedoch resultierte daraus ein Ciphertext von 128 Byte. Zusammen mit der Signatur entstehen somit 256 Byte an zu versendenden Daten. Da die vorliegenden Smart- Cards jedoch nur 255 Byte an Daten unterstützen, wurde sich für eine Reduzierung der Schlüssellänge entschieden. Das Cryptography-Applet stellt folgende öffentlich zugängliche Anweisungen bereit:

Wie an den Anweisungsnamen erkennbar ist, ist es möglich, das Schlüsselpaar bestehend aus privaten und öffentlichen Schlüssel für die Karte zu importieren. Dies ist notwendig, da sonst die Signierung nicht als Sicher eingestuft werden kann. Nach der Installation der Applets befindet sich die Karte in ihrem Werkszustand. Es sind keine Schlüsselpaare und auch keine Daten auf der Karte gesetzt. Um die Karte benutzen zu können, müssen nun als erstes die Schlüssel für das RSA Kryptosystem gesetzt werden. Dazu wird die OffCard-Anwendung genutzt. Eine nachträgliche Änderung der Schlüssel wird mit Hilfe von Flags unterbunden. Die Karte ist somit gebrandmarkt. Im gesamten Hotel existiert ein Schlüsselpaar für die Karten und ein Schlüsselpaar für die Terminals. Auch wenn eine dritte Partei eine Karte im Werkszustand in die Hand bekommen und seine eigenen Schlüssel setzen sollte bleibt das System sicher. Es ist nicht möglich, an die Karte gesendete

Daten zu entschlüsseln, da der private Schlüssel der Karte falsch ist sowie die Signatur nicht mit dem privaten Schlüssel der Terminals verifiziert werden kann. Aufgrund der nicht passenden Schlüssel ist es ebenso wenig möglich, gefälschte Daten an die OffCard-Anwendung zu schicken. Das System wird erst unsicher, wenn die Schlüsselpaare für Karten und Terminals bekannt würden. Mit ihnen ist es dann möglich vertrauenswürdige Karten zu fälschen. Die Methoden für die Ver- und Entschlüsselung sind innerhalb der Karte über die Applet-Firewall zugänglich. Den Applets Student und Disco ist es erlaubt, eine Instanz des Crypto-Applets zu erhalten. Je nach Richtung der Datenübertragung können diese Applets dann entweder Daten ver- oder entschlüsseln.

Beim Aufruf der Entschlüsselungs-Methode (decrypt) werden die 64 Bit Daten mit dem privaten Schlüssel der Karte entschlüsselt. Der dadurch gewonnene Klartext wird mithilfe des öffentlichen Schlüssels des Terminals und der mitgesendeten Signatur verifiziert. Der Klartext wird für die weitere Verwendung im Puffer abgelegt. In diesem Fall ist davon auszugehen, dass die Daten manipuliert wurden. Beim Aufruf der Verschlüsslungs-Methode (encrypt) werden die in die Methode übergebenen Daten mit dem privaten Schlüssel der Karte signiert. Weiterhin werden die Daten mit dem öffentlichen Schlüssel des Terminals verschlüsselt. Daten und Signatur werden im Puffer abgelegt und können vom aufrufenden Applet versendet werden. Im Gegensatz zu allen anderen Applets ist für das Crypto-Applet keine Reset Möglichkeit vorgesehen. Um die Schlüssel neu setzen zu können, muss das Applet neu installiert werden.

# 5 OffCard

## 5.1 Simulation

Für die Benutzung der OffCard-Anwendung mit der simulierten SmartCard muss in der opencard.properties Datei die Konfiguration für die Simulation aktiv sein. Die Konfiguration für die reale SmartCard muss mit Zeilenkommentaren (#) deaktiviert werden. Die Simulation der SmartCard muss im Eclipse gestartet werden.

Anschließend muss das Terminal mit dem Port 8050 geöffnet werden. Dazu kann folgender Befehl verwendet werden: /terminal "Remote|localhost:8050"Nach dem Freigeben der Verbindung mit /close kann die OffCard-Anwendung gestartet und benutzt werden.

## 5.2 Connection-Tab

todo

## 5.3 Configuration-Tab

todo

## 5.4 Student-Tab

todo

## 5.5 Disco-Tab

# 6 Fazit

todo