Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

Prüfung von öffentlichen eID-Terminals mit einem Android-Smartphone

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)

eingereicht von: Ole Richter
geboren am: 29.09.1991
geboren in: Forst (Lausitz)

Gutachter/innen: Prof. Dr. rer. nat. Jens-Peter Redlich
Prof. Dr. rer. nat. Ernst-Günter Giessmann

eingereicht am: ..... verteidigt am: .....

Inhaltsverzeichnis

Abkürzungen

AID	Application Identifier
APDU	Application Protocol Data Unit
ASN.1	Abstract Syntax Notation One
ATR	Answer To Reset
BER	Basic Encoding Rules
CA	Chip Authentication
DG	Data Group
CHAT	Card Holder Authorization Template
EAC	Extended Access Control Version 2
eID	electronic Idenfification
HCE	Host-based Card Emulation
MF	Master File
NFC	Near Field Communication
nPA	neuer Personalausweis
PACE	Password Authenticated Connection Establishment
PIN	Persönliche Identifikationsnummer
PUC	PIN Unblocking Key
TA	Terminal Authentication
TLV	Tag-Length-Value
VfB	Vergabestelle für Berechtigungszertifikate

1 Einleitung

Seit November 2010 haben Bürgerinnen und Bürger der Bundesrepublik Deutschland die Möglichkeit, die eID-Funktion des neuen Personalausweises (nPA) zum Identitätsnachweis zu nutzen. Ähnlich dem Vorzeigen des Personalausweises als Sichtdokument ist es so möglich, sich elektronisch gegenüber Behörden und zertifizierten Dienstanbietern auszuweisen.

Damit der Ausweisinhaber bei der Authentifizierung weiß, welcher Dienst welche Daten auf dem Ausweis abfragt, werden ihm diese Informationen vor der Übermittlung der Daten an den Dienst angezeigt. Dabei muss er allerdings den Angaben, die ihm vom Terminal angezeigt werden vertrauen. Dieses Vertrauen ist in der Regel berechtigt, solange sich das Terminal im Besitz des Ausweisinhabers befindet. Möglich sind aber auch Szenarien, in denen sich das Terminal nicht im Besitz des Ausweisinhabers befindet, wie zum Beispiel an öffentlichen Terminals. In diesem Szenario muss der Ausweisinhaber dem fremden Terminal vertrauen und kann zunächst nicht überprüfen, ob auch genau die zur Preisgabe bestimmten Daten abgefragt werden. Ein Terminal könnte sich nun zu einem anderen, nicht dem angezeigten Dienst entsprechenden Dienst verbinden oder die Angaben über die abzufragenden Daten verfälschen. In diesem Fall würde der Ausweisinhaber die Kontrolle über die auf seinem Personalausweis gespeicherten Daten verlieren.

Um ein Terminal auf Vertrauenswürdigkeit zu prüfen, schlug Frank Morgner in seiner Arbeit "Mobiler Chipkartenleser für den neuen Personalausweis" vor, den nPA für eine Prüfung des Terminals zu emulieren [9, S. 80.].

So kann vor der eigentlichen Kommunikation zwischen dem Terminal und dem nPA der Emulator mit dem Terminal kommunizieren und dem Ausweisinhaber die entsprechenden Berechtigungen und die Zertifikatsbeschreibung des Terminals anzeigen. Nach der Prüfung durch das Anzeigen der Informationen kann dann die Authentifizierung mit dem echten nPA erfolgen.

Ziel dieser Arbeit ist es, die Idee von Frank Morgner umzusetzen und einen solchen Emulator für das mobile Betriebssystem Android zu implementieren. Auch werden die Voraussetzungen untersucht, die das Android-Gerät besitzen muss, um den nPA ausreichend für die Prüfung eines Terminals zu emulieren.

Die Arbeit ist in drei Abschnitte gegliedert. Im ersten Abschnitt wird auf den elektronischen Personalausweis, die eID-Anwendung und die technischen Grundlagen eingegangen. Der zweite Teil beschäftigt sich mit der Problemstellung und den theoretischen Aspekten der Umsetzung zur Prüfung eines Terminals. Im dritten und letzten Abschnitt geht es um die konkrete Implementierung einer Emulation des nPA.

2 Der elektronische Personalausweis

Der elektronische Personalausweis (nPA) stellt neben der Funktion als Sichtdokument verschiedene elektronische Dienste zur Verfügung. Es gibt die hoheitliche Funktion, die es Behörden, wie dem Zoll, erlaubt, die Daten auf dem nPA auszulesen. Diese Funktion ist ähnlich dem des ePass. Des weiteren gibt es nicht-hoheitliche Funktionen, die privat genutzt werden können. Dies sind die eID-Anwendung, die zur elektronischen Identifizierung und Authentisierung gegenüber bestimmten Diensten dient und die Funktion zur Erstellung qualifizierter elektronischer Signaturen (QES) durch den nPA. Für die vorliegende Arbeit ist im Besonderen die eID-Anwendung von Bedeutung, da diese im Zusammenhang mit öffentlichen Terminals genutzt werden kann.

Für die elektronische Nutzung enthält der nPA einen RFID-Chip, mit dem die kontaktlose Kommunikation über NFC möglich ist. Die Kommunikation findet gemäß dem ISO/IEC 14443 Standard [7] statt. Das Datenformat für die Kommunikation zwischen dem nPA und dem Kartenleser wird durch den Standard ISO-7816 [8] festgelegt.

2.1 Die eID-Anwendung des nPA

Die Authentifizierung mithilfe der eID-Anwendung erlaubt es dem Ausweisinhaber, eGovernment- und eBusiness-Dienste [4, Seite 13] zu nutzen. Die einzelnen Dienste sind über das Internet nutzbar. Bei der Authentisierung werden vom Terminal einzeln die benötigten Informationen vom nPA abgefragt. Zum Beispiel könnte das Terminal den Namen des Ausweisinhabers oder dessen Wohnort abfragen. Dadurch, dass die Daten einzeln abgefragt werden, ist es möglich, nur die benötigten Informationen an das Terminal zu übertragen. Soll zum Beispiel ein Altersnachweis erbracht werden, so ist es nicht nötig und im Sinne des Datenschutzes auch nicht gewollt, dass weitere Informationen über den Ausweisinhaber übertragen werden. Die verschiedenen Daten, die aus dem nPA ausgelesen werden können, sind in Abbildung 6 einzeln dargestellt.

Im Gegensatz zur eID-Anwendung des ePass werden die abgefragten Daten nicht signiert. So kann die Echtheit der Daten später nicht gegenüber Dritten nachgewiesen werden. Dies soll den lukrativen Verkauf der Daten verhindern.

In Abbildung 1 ist die Kommunikation zwischen den einzelnen Kommunikationspartnern während der eID-Nutzung dargestellt. Der nPA kommuniziert direkt über die Luftschnittstelle mit dem Kartenleser und somit mit dem Local Terminal. Zwischen dem Local Terminal und dem Remote Terminal wird zunächst eine sichere Verbindung aufgebaut (SSL/TLS), über die dann später, nach erfolgreicher EAC, der Secure-Messaging-Kanal zwischen dem nPA und dem Remote Terminal für die Übertragung der geschützten Daten zur Verfügung steht.

2.2 Kommunikation zwischen Chipkarte und Kartenleser

Die Kommunikation zwischen der Chipkarte (in diesem Fall der nPA) und dem Chipkartenleser ist im Standard ISO-7816 festgeschrieben. Um den nPA zu emulieren muss also dieser Standard in gewissen Punkten eingehalten werden. Bei der Implementierung

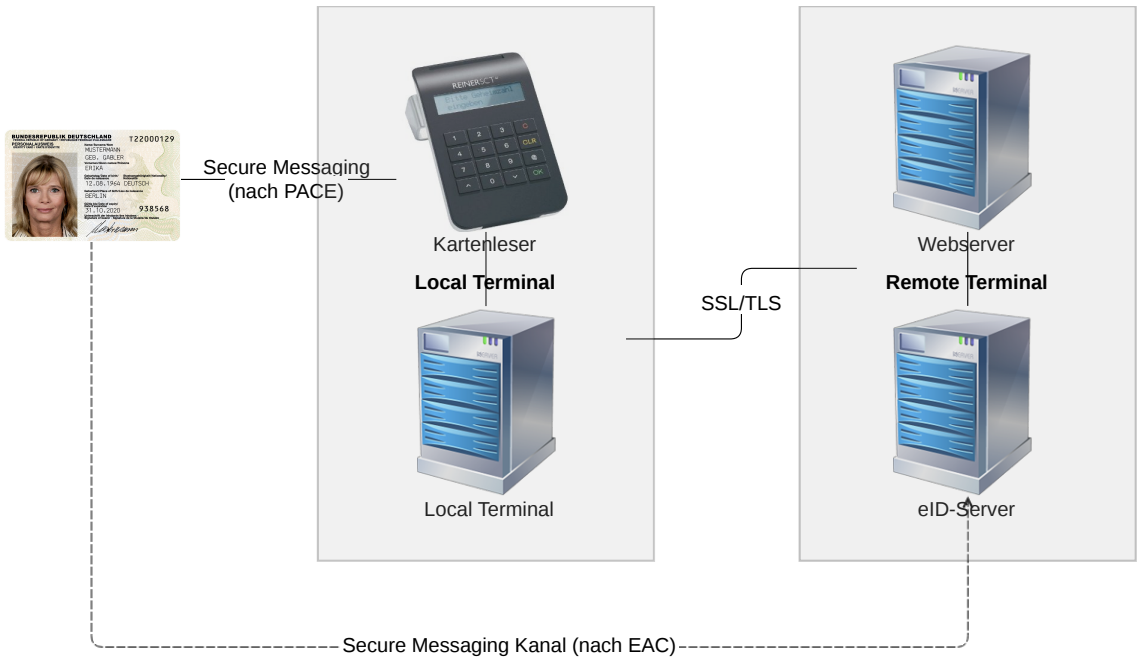

Abbildung 1: Vereinfachte Darstellung der Kommunikation bei der eID-Anwendung.

reicht es, sich auf die Anwendungsschicht zu beschränken, da die darunter liegenden Schichten bereits gemäß dem Standard durch das Betriebssystem Android umgesetzt sind.

Die kleinste Einheit in der paketorientierten Anwendungsschicht ist die APDU (Application Protocol Data Unit), die sowohl Daten als auch Protokoll-Steuerinformationen enthält. Leser und Karte arbeiten dabei mit zwei unterschiedlich aufgebauten APDUs. Der Leser schickt Kommandos in Form von Command APDUs an die Karte und diese antwortet in Form von Response APDUs.

Command APDU: Die Command APDU setzt sich aus einem Header und einem Body zusammen. Im Header werden die Befehlsklasse (CLA), das eigentliche Kommando (INS) und zwei Parameter (P1 und P2) angegeben. Der optionale Body enthält Kommandodaten und Angaben über die Länge der Daten (L_c), sowie über die Länge der erwarteten Antwortdaten (L_e).

CLA	INS	P1	P2	L_c	Daten-Feld	L_c
Header				Body		

Abbildung 2: Aufbau Command APDU

Response APDU: Die Response APDU besteht aus einem optionalen Body, in welchem die Antwortdaten stehen und einem Trailer, welcher zwei Statusbytes (SW1

und SW2) enthält. In den Statusbytes wird der Status der Abarbeitung des zuvor gesendeten Kommandos kodiert.

Daten-Feld	SW1	SW2
Body	Trailer	

Abbildung 3: Aufbau Response APDU

Für den umzusetzenden Anwendungsfall reicht es, nur einige bestimmte Kommandos zu implementieren. Dabei handelt es sich um die Kommandos "Select File" und "Read Binary". Das Kommando "Select File" dient der Auswahl von Dateien auf der Chipkarte (in diesem Fall der nPA) und mit dem Kommando "Read Binary" werden Dateien mit einer transparenten Struktur ausgelesen, wie zum Beispiel die Datei EF.CardAccess.

Diese beiden Kommandos werden vom Terminal benötigt, um auf die Datei EF.CardAccess zuzugreifen, welche relevante SecurityInfos enthält. Für den weiteren Protokolldurchlauf sind weitere Kommandos notwendig, die in den späteren Kapiteln erläutert werden.

Da die Größe des Datenfeldes nur in einem Byte kodiert werden kann, ist sie nach oben hin beschränkt. Fallen größere Datenmengen an, können diese entweder auf mehrere APDUs aufgeteilt oder mit extended length APDUs übertragen werden. Die extended length APDUs müssen allerdings auch von beiden Seiten - dem Kartenleser und der Karte - unterstützt werden. Diese Unterstützung ist zum jetzigen Zeitpunkt bei vielen Android-Geräten nicht gegeben.

2.3 Die Datei EF.CardAccess

Die Datei EF.CardAccess enthält relevante SecurityInfos, die für EAC benötigt werden. Sie ist auf der Chipkarte gespeichert und kann vom Terminal abgefragt werden. Die Informationen liegen als DER-kodierte ASN1-Datenstruktur vor. Die Informationen werden vom Terminal abgefragt, um für die Protokolle die richtigen Algorithmen und Parameter auszuwählen. Folgende SecurityInfos befinden sich in der Datei: [4, S. 16]

- PACEInfo
- PACEDomainParameterInfo
- ChipAuthenticationInfo
- ChipAuthenticationDomainParameterInfo
- TerminalAuthenticationInfo
- CardInfoLocator

Um die Datei auszulesen, muss sie zunächst vom Terminal mit einem "Select File" Kommando ausgewählt werden. Ist die Datei größer als 256 Byte, wird sie vom Terminal in mehreren Schritten ausgelesen. Für das Auslesen ist keine Authentifizierung des Terminals erforderlich.

2.4 Extended Access Control

Extended Access Control (EAC) [1] vereinigt mehrere Protokolle, die die Vertraulichkeit, Integrität und Authentizität der Kommunikation mit dem Ausweis sicherstellen sollen. Um die Kommunikation zu verstehen, werden drei verschiedene Instanzen definiert: der nPA, das Local Terminal und das Remote Terminal. Der nPA ist das Ausweisdokument, welches sich im Besitz des Ausweisinhabers befindet. Das Local Terminal ist das Terminal, welches direkt mit dem nPA kommuniziert. Es befindet sich in physischer Nähe zum nPA und verbindet sich mit diesem über die Luftschnittstelle. Das Remote Terminal bezeichnet den Dienst, gegenüber dem sich der Ausweisinhaber authentisieren will. Das Remote Terminal muss sich nicht in physischer Nähe zum nPA befinden, es genügt eine Verbindung über das Internet. Die Kommunikation über die Luftschnittstelle zwischen nPA und Local Terminal wird mit dem Protokoll PACE abgesichert. Die Kommunikation zwischen dem nPA und dem Remote Terminal wird im weiteren Durchlauf mit den Protokollen TA, Passive Authentisierung und CA abgesichert. Noch bevor der PACE-Durchlauf gestartet wird, sendet das Terminal die Zugriffsrechte für den Durchlauf an den nPA. Diese Zugriffsrechte sind im sogenannten CHAT kodiert, welcher in Kapitel 2.6 genauer beschrieben wird.

Die Kommunikation zwischen dem nPA, dem Local Terminal und dem Remote Terminal ist in Abbildung 4 vereinfacht dargestellt.

2.4.1 PACE

Das Protokoll Password Authenticated Connection Establishment (PACE) dient zur passwortbasierten Authentisierung und Schlüsselaushandlung. Es sichert nicht nur die Luftschnittstelle zwischen dem Local Terminal und dem nPA ab, sondern gewährleistet auch die willentliche Authentifizierung des Ausweisinhabers gegenüber dem nPA. Bei der Authentifizierung gibt der Ausweisinhaber das entsprechende Geheimnis (PIN oder CAN) am PIN Pad des Local Terminals ein. Nach dem erfolgreichen PACE-Durchlauf werden die über die Luftschnittstelle übertragenen Daten durch Secure Messaging abgesichert. Das Local Terminal muss neben einem RFID-Kartenleser auch über ein Display (Dies kann auch ein PC, auf dem die AusweisApp läuft und ein Basiskartenleser sein) zum Anzeigen der Berechtigungen und Informationen zum Dienstanbieter (Name, Anschrift und Zweck des Auslesens) und über ein PIN Pad zur Eingabe des Geheimnisses durch den Ausweisinhaber verfügen.

Die Geheimnisse, die für PACE verwendet werden können sind die eID-PIN (Geheimzahl für die eID-Anwendung), die PUK (PIN Unblocking Key), die MRZ (Maschinenlesbare Zone) und die CAN (Kartenzugriffsnummer). Für die eID-Anwendung und die Prüfung von Terminals ist vor allem die eID-PIN interessant. Diese sollte nur dem Ausweisinhaber bekannt sein, damit sich nur dieser gegenüber dem nPA authentifizieren kann. Eine Änderung der eID-PIN ist mit entsprechender Client-Software unter der Eingabe der alten eID-PIN jederzeit möglich. Nach Fehleingaben der PIN kann

[1] in dieser Arbeit ist immer Extended Access Control Version 2 gemeint

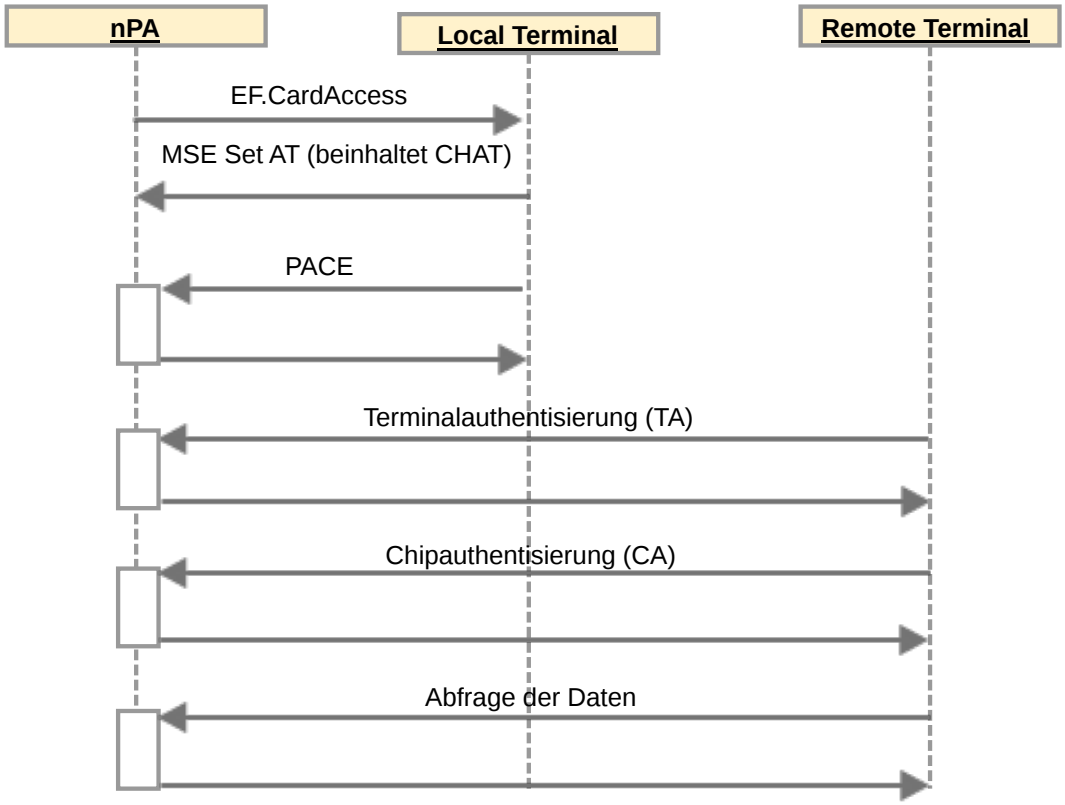

Abbildung 4: Vereinfachte Darstellung der Kommunikation zwischen dem nPA, dem Local Terminal und dem Remote Terminal bei der eID-Anwendung durch ein Sequenzdiagramm.

der Fehlbedinungszähler mit der PUK (bei drei Fehlversuchen) wieder zurück gesetzt werden.

Der PACE-Durchlauf besteht aus folgenden vier Schritten, nach denen die zwei Sitzungsschlüssel K_{Enc} zur Verschlüsselung und K_{Mac} zur Integritätssicherung zur Verfügung stehen.

1. Der nPA erzeugt eine Zufallszahl (Nonce) und verschlüsselt diese mit einem aus dem Geheimnis abgeleiteten Schlüssel. Die verschlüsselte Nonce wird dann an das Terminal gesendet. Das Terminal entschlüsselt daraufhin die Nonce mit dem ebenfalls bekannten Geheimnis.

2. Zum Berechnen der Diffie-Hellman Domain Parameter werden weitere Daten ausgetauscht. Mit einer Mapping-Funktion werden die Daten auf einen Erzeuger einer mathematischen Gruppe abgebildet.

3. Es wird ein Diffie-Hellman-Schlüsselaustausch mit dem zuvor generierten Erzeuger durchgeführt. Aus dem entstandenen gemeinsamen Geheimnis K werden die beiden Sitzungsschlüssel K_{Enc} und K_{Mac} abgeleitet.

4. Zur gegenseitigen Authentisierung wird eine Prüfsumme des gerade erzeugten Schlüssels K_{Mac} erzeugt und ausgetauscht.

2.4.2 Secure Messaging

Nach der Aushandlung der Sitzungsschlüssel durch PACE werden die über die Luftschnittstelle zwischen dem nPA und dem Local Terminal zu übertragenden Daten sicher durch Secure Messaging (SM) übertragen. Die Daten werden dabei Verschlüsselt um die Vertraulichkeit zu gewährleisten und mit einer Checksumme (MAC) versehen, um auch die Integrität der Nachrichten sicher zu stellen. Durch einen Zähler wird das Wiedereinspielen von Nachrichten verhindert.

Die APDUs, die über Secure Messaging verschickt werden, folgen einem bestimmten Aufbau. Sie bestehen aus einzelnen BER-TLV-kodierten Objekten, die in Abbildung 5 angegeben sind.

Tag	**Beschreibung**
0x87	Padding-content indicator followed by cryptogram
0x97	Protected Le
0x99	Processing Status
0x8E	Cryptographic Checksum

Abbildung 5: Secure Messaging Data Objects [12, S. 73]

Bei der Implementierung von Secure Messaging müssen die Objekte gemäß der Technischen Richtlinie BSI TR-03110-3 [12, S. 73 ff.] erstellt und verarbeitet werden. Ob eine Nachricht über Secure Messaging verschickt wird, lässt sich am Class Byte ablesen.

2.4.3 TA, Passive Authentisierung und CA

Wurde durch PACE ein sicherer Kanal zwischen dem nPA und dem Local Terminal aufgebaut, erfolgen die Authentisierung des Remote Terminals und die Authentisierung des nPA.

Zuerst muss das Terminal bzw. der eID-Dienst durch die Terminal Authentisierung (TA) seine Berechtigungen nachweisen. Dafür existiert die dreistufige EAC-PKI-Struktur, durch die der nPA die Berechtigungen des eID-Dienstes überprüfen kann. Diese Berechtigungen werden durch die Vergabestelle für Berechtigungszertifikate (VfB) erteilt. Um zu gewährleisten, dass möglichst wenige Daten durch den Ausweisinhaber preis gegeben werden, werden einzelne Berechtigungen nur vergeben, wenn dafür eine Notwendigkeit besteht.[13, S. 43 ff.] Bei der TA wird zunächst die benötigte Zertifikatskette (bis zum Vertrauensanker, der auf dem nPA gespeichert ist) an den nPA übertragen. Es können also dem Nutzer der Emulation Informationen zu den Zertifikaten angezeigt werden. Der eigentliche Nachweis erfolgt dann über ein Challenge-Response-Verfahren. Es wird ein Hash über die bei PACE genutzte Nonce, den öffentlichen PACE-Schlüssel und den öffentlichen CA-Schlüssel gebildet und mit dem öffentlichen Schlüssel des eID-Services verschlüsselt. Der eID-Service schickt nun den entschlüsselten Hash an den nPA und weist somit den Besitz des privaten Schlüssels nach, der zum entschlüsseln benötigt wird. An dieser Stelle werden die TA, die CA, PACE und somit auch die Berechtigungen aus dem CHAT aneinander gebunden.

Nach der erfolgreichen TA erfolgt die Chipauthentiserung (CA). Diese dient dazu, dem Terminal die Echtheit des nPA nachzuweisen und eine sichere Verbindung zwischen dem nPA und dem Remote Terminal aufzubauen. Der nPA besitzt für die CA ein Schlüsselpaar, bestehend aus einem öffentlichen und einem privaten Schlüssel. Der öffentliche Schlüssel ist durch die Dokumenten-PKI signiert und wird zusammen mit einer Zufallszahl an das Terminal gesendet. Der private Schlüssel ist auf dem Chip gespeichert und gegen Auslesen geschützt. Das Terminal erzeugt ebenfalls ein Schlüsselpaar und schickt den öffentlichen Schlüssel an den Chip. Nun berechnen beide Seiten ein gemeinsames Geheimnis, aus dem ähnlich wie bei PACE Sitzungsschlüssel abgeleitet werden,. Mit den Sitzungsschlüsseln wird die weitere Kommunikation abgesichert. Die Signatur des öffentlichen Schlüssels des nPA wird vom Terminal bei der Passiven Authentisierung abgefragt. Somit hat der nPA den Besitz des privaten Schlüssels und damit auch seine Echtheit nachgewiesen. [5]

2.5 Terminal-Zertifikate

Die Berechtigungszertifikate sind CV-Zertifikate (Card Verifiable Certificates). Neben dem Signaturalgorithmus, der Domänenparameter und dem öffentlichen Schlüssel sind auch die Rollenzuweisung und die möglichen Berechtigungen angegeben. Um die Aktualität des Zertifikats prüfen zu können, enthält es ebenfalls das Ausstellungsdatum, sowie das Ablaufdatum. Die Zertifikate werden nach eingehender Überprüfung des Dienstes durch die Vergabestelle für Berechtigungszertifikate (VfB) vergeben [4].

Zur Überprüfung der Zertifikate durch den nPA gibt es eine dreistufige Berechtigungs-

PKI, die sogenannte EAC-PKI [12], die folgendermaßen aufgebaut ist: An oberster Stelle steht die Country Verifying Certification Authority (CVCA) als Wurzelinstanz. Darunter mehrere Document Verifier (DV), jeweils für zusammenhängende Terminals. Die dritte und unterste Stufe stellen die einzelnen Terminals (Inspektionssysteme, Authentisierungsterminals und Signaturterminals) dar. Da kein Rückruf der Zertifikate vorgesehen ist, werden Zertifikate nur mit kurzer Laufzeit ausgestellt (ausgenommen sind Zertifikate für Signaturterminals). Die entstehende Zertifikatskette wird, wenn benötigt, bei der TA zum nPA übertragen.

Neben dem Terminaltyp und den Zugriffsrechten stehen im Zertifikat für die Authentisierungsterminals auch Angaben zum Dienstanbieter. Diese Angaben werden durch die Certificate Extension Certificate Description realisiert [4, S. 39]. Es gibt folgende Angaben, die dem Nutzer angezeigt werden sollen, noch bevor er das Geheimnis (PIN oder PUK) für den PACE-Durchlauf angibt:

- Name der ausstellenden Berechtigungs-CA (issuerName);
- Internetadresse des Ausstellers (issuerURL)
- Name des Dienstanbieters (subjectName);
- Internetadresse des Dienstanbieters (SubjectURL)
- Zweck des Auslesevorgangs (termsOfUsage)
- Weitere optionale Angaben

Der nPA erhält während der TA allerdings nur einen Hash über die Certificate Description und nicht die wirklichen Angaben zum Dienstanbieter. Soll dem Benutzer der Emulation die Zertifikatsbeschreibung angezeigt werden, könnte man die Zertifikatsbeschreibungen und die dazugehörigen Hashwerte in einer Datenbank hinterlegen. Diese Datenbank ist momentan noch nicht Teil des Emulators. Wie schon bei Frank Morgner erwähnt, könnte eine solche Datenbank von öffentlicher Stelle angeboten werden [9, S. 81].

2.6 Certificate Holder Authorization Template (CHAT)

Um die Zugriffsrechte, die ein Terminal besitzt anzuzeigen, ist es notwendig das Certificate Holder Authorization Template (CHAT) zu parsen. In diesem sind die einzelnen Zugriffsrechte kodiert.

Das erste Kommando, welches bei der Initialisierung von PACE vom Terminal geschickt wird ist das Kommando MSE:Set AT. In diesem Kommando ist unter anderem der CHAT enthalten. Im CHAT stehen die Rolle des Terminals und die Berechtigungen zum Auslesen der einzelnen Datengruppen, sowie die angeforderten Berechtigungen für einzelne Funktionen, die aufgerufen werden sollen (siehe Abbildung 6). Diese Informationen sollen dem Ausweisinhaber zusammen mit der Zertifikatsbeschreibung angezeigt werden, bevor dieser das für PACE notwendige Geheimnis angibt und dem

Terminal damit den Zugriff auf die Daten gestattet. Der CHAT wird an die Initialisierung von PACE gebunden, sodass im weiteren Verlauf nur die angegebenen Rechte verwendet werden können.

Es werden vier verschiedene Terminaltypen unterschieden: Inspektionssysteme, Authentisierungsterminals, Bestätigte Signatureterminals und Nicht authentisierte Terminals. Die Beschreibung der Teminals wird sich jedoch auf die auf Authentisierungsterminals beschränken, da die anderen Terminaltypen für die eID-Anwendung nicht relevant sind.

Die Berechtigung sowie die Rolle des Terminals sind in 5 Byte kodiert, die als Bitmap zu interpretieren sind. Die ersten zwei Bit geben an, welche der vier Rollen das Terminal einnimmt. Die restlichen Bit stehen jeweils für eine Berechtigung des Terminals. Unterteilt sind die Berechtigung in Schreib-Rechte, Lese-Rechte und Rechte für spezielle Funktionen. Mit diesen Berechtigung können bestimmte Daten auf dem Ausweis abgerufen werden. Diese einzelnen Daten, auf die durch die Berechtigungen zugegriffen werden kann, sind in Abbildung 6 im Detail dargestellt. [12, S. 63 f.] Im CHAT nicht enthalten sind Hinweise auf das entsprechende Zertifikat des Terminals oder der Name des eID-Services. Dieses wird im Protokollverlauf von PACE erst später während der TA angegeben.

Datei	**Inhalt**	**mögliche Berechtigungen**
DG1	Dokumententyp	Lese-Berechtigung
DG2	Ausgebender Staat	Lese-Berechtigung
DG3	Ablaufdatum	Lese-Berechtigung
DG4	Vorname(n)	Lese-Berechtigung
DG5	Familienname	Lese-Berechtigung
DG6	Ordensname / Künstlername	Lese-Berechtigung
DG7	Doktorgrad	Lese-Berechtigung
DG8	Geburtsdatum	Lese-Berechtigung
DG9	Geburtsort	Lese-Berechtigung
DG10	Staatsangehörigkeit	Lese-Berechtigung
DG11 - DG12	unbenutzt	-
DG13	Geburtsname	Lese-Berechtigung
DG14 - DG16	unbenutzt	Lese-Berechtigung
DG17	Adresse	Lese- und Schreib-Berechtigung
DG18	Wohnort-ID	Lese- und Schreib-Berechtigung
DG19	Nebenbestimmungen I	Lese- und Schreib-Berechtigung
DG20	Nebenbestimmungen II	Lese- und Schreib-Berechtigung
DG21	Unbenutzt	-
-	Vergleichsgeburtssdatum für Altersverifikation	-
-	Schlüssel für dienstanbieterspezifisches Sperrmerkmal	-
-	Schlüssel für dienst- und kartenspezifische Kennung	-

Abbildung 6: Dateiengruppen auf dem nPA und die jeweils möglichen Berechtigungen

3 Problemdarstellung

Bei der Entwicklung eines Emulators, der möglichst an allen Terminals kompatibel einsetzbar ist und nicht oder nur sehr schwer von einem echten nPA zu unterscheiden ist, ergeben sich einige Schwierigkeiten.

Prinzipiell ist es nicht ohne weiteres möglich, den nPA komplett zu emulieren. Spätestens bei der CA, bei der sich der nPA gegenüber dem eID-Dienst authentisieren muss, wird die Emulation ohne die entsprechenden geheimen CA-Schlüssel als solche erkannt [9]. Für die Überprüfung des Terminals ist dies zunächst jedoch nicht nötig, da sich das Terminal schon vorher während der TA authentisieren muss. Ebenfalls vor der CA legt sich das Terminal auch auf die Berechtigungen für die Abfrage der Daten fest. Folglich genügt es, entsprechende, zum Standard ISO 7816 konforme Hardware zu verwenden und alle Schritte bis einschließlich der TA in Software zu emulieren [9]. Um nur die Berechtigungen auszulesen wird nicht einmal die Implementierung von PACE oder der TA benötigt. Die APDU, die den CHAT und somit auch die Berechtigungen enthält wird bereits bei der Initialisierung von PACE unverschlüsselt übertragen. Allerdings kann so nicht zweifelsfrei festgestellt werden, ob das vom Terminal verwendete Zertifikat echt ist.

Damit die Emulation als Ausweis erkannt wird, muss eine korrekte EF.CardAccess von der Emulation bereitgestellt und übertragen werden. Diese könnte von einem echten nPA übernommen oder gemäß TR-03110-3 erstellt werden[9].

3.1 Schwachstellen der Prüfung durch eine Emulation

Wird bei der Kommunikation der Emulation des nPA nicht die komplette TA durchgeführt, womit der eID-Dienst seine Echtheit nachweisen müsste, könnte ein böswilliges Terminal die Prüfung verhindern. In diesem Fall könnte das böswillige Terminal bei der Kommunikation mit der Emulation im CHAT nur die Berechtigungen angeben, die auch dem Benutzer des Terminals angezeigt werden. Es könnte auch ein beliebiges, dem Benutzer vertrauenswürdig erscheinendes Zertifikat an die Emulation übertragen. Wird dann nach der Überprüfung durch die Emulation der echte nPA zur eigentlichen Authentisierung genutzt, könnte das Terminal ungehindert die vom Nutzer nicht gewollte Authentisierung durchführen.

Wird die TA von der Emulation komplett durchgeführt und die Echtheit des Terminals überprüft, ist das Terminal nicht mehr in der Lage, sich als anderen Dienst auszugeben. Die Berechtigungen, die vor dem PACE-Durchlauf im CHAT an den nPA übertragen werden, könnten in diesem Fall nur noch begrenzt verfälscht werden. Es wäre nur eine Untermenge der Berechtigungen, die im Zertifikat angeführt werden bei der Angabe möglich. Werden andere Berechtigungen angegeben, würde die Unstimmigkeit bei der Prüfung der Berechtigungen im CHAT und des Zertifikats auffallen.

Eine Erkennung der Emulation durch das Terminal kann durch mehrere Unterschiede zum echten nPA erfolgen: Fehler in der Implementierung, ein zum nPA unterschiedlicher ATR am Anfang der Kommunikation, Kameras, die feststellen, dass sich kein nPA am Lesegerät befindet oder Unterschiede in der Geschwindigkeit, mit der die Emulation

antwortet.

Selbst wenn es dem Terminal nicht gelingt, die Emulation als solche zu erkennen, könnte zufällig zwischen der vom Nutzer nicht gewollten Authentisierung und der vom Nutzer erwarteten Authentisierung ausgewählt werden. Hierbei wäre eine vorangegangene Prüfung des Terminals nicht zwangsläufig korrekt.

3.2 Alternative Möglichkeiten zur Überprüfung eines Terminals

Neben der Emulation des NPA auf einem Android-Smartphone sind auch andere Methoden denkbar, ein Terminal zu überprüfen.

Eine von Frank Morgner angeführte Alternative besteht darin, eine Weiterleitung zu bauen, mit der die Kommunikation zwischen NPA und Leser gelesen und wenn nötig auch verändert werden kann [9]. Dies würde einer Firewall zwischen Terminal und nPA entsprechen. Der Vorteil dieser Variante besteht darin, dass der NPA nicht emuliert werden muss. Allerdings werden zwei Verschiedene NFC-fähige Geräte für die Weiterleitung benötigt. Dabei würde das eine Gerät mit dem Ausweis kommunizieren und das andere mit dem Terminal. Erfolgreich durchgeführt wurde eine solche Weiterleitung bereits mit dem Tool pcsc-relay [9, S. 114].

Eine andere Möglichkeit besteht darin, die Softwarekomponenten auf einen Server, der mit dem NFC-Device kommuniziert, auszulagern. so würde keine Notwendigkeit bestehen, die Softwarekomponenten auf einem Android-Gerät zu implementieren. Diese Lösung hätte allerdings den Nachteil, dass eine Server-Architektur zur Verfügung stehen müsste und das NFC-Device auf eine Internetverbindung angewiesen wäre.

4 Implementierung

Die Implementierung erfolgt auf einem mobilen Android-Gerät, auf dem der nPA teilweise emuliert wird. Am Ende soll durch die Emulation eine, für den Anwender leicht durchzuführende Prüfung eines eID-Terminals ermöglicht werden.

Als Basis für die Implementierung wurde das weit verbreitete, mobile Betriebssystem Android gewählt. Vorausgesetzt ist also ein mobiles Android-Gerät mit NFC-Schnittstelle (es gibt allerdings noch weitere Anforderungen, auf die später eingegangen wird).

Die Funktionalität der App beschränkt sich auf die teilweise Emulation eines nPA und das Anzeigen der relevanten Informationen, die für die Prüfung eines Terminals notwendig sind. Die Informationen werden dabei in einer für den Nutzer möglichst einfach zu lesenden Form mit einer graphischen Benutzeroberfläche angezeigt. Zur Prüfung muss der Anwender nichts weiter tun, als die App zu starten und das Android-Gerät an das Lesegerät des Terminals halten. Die Informationen werden dann automatisch auf der graphischen Oberfläche angezeigt.

Noch nicht implementiert wurde die komplette Durchführung der TA und damit die Verifizierung der Echtheit des angegebenen eID-Services. Dies war mit der eingesetzten Hardware aufgrund der eingeschränkten Verarbeitung von extended length APDUs durch Android-Geräte im Kartenemulations-Modus nicht ohne weiteres möglich. Allerdings ließe sich die Software leicht um die komplette Durchführung der TA erweitern. Das Anzeigen der Informationen zu den Zertifikaten, die während der TA übertragen werden ist zwar implementiert, allerdings wird hierbei im Allgemeinen Hardware benötigt, die extended length APDUs verarbeiten kann, da die meisten Zertifikate eine Länge von 255 Bytes überschreiten und vom Terminal in extended length APDUs an den nPA verschickt werden.

4.1 Entwicklungsumgebung

Als Entwicklungsumgebung kam die integrierte Entwicklungsumgebung Eclipse zusammen mit dem Android Development Tools (ADT) zum Einsatz. Um die Entwicklung unkompliziert zu halten, wurde der Quelltext in Java geschrieben. Für das Verarbeiten der ASN.1-Strukturen wurde der Cryptography Service Provider Spongycastle verwendet. Bei der Implementierung des PACE-Protokolls und von Secure Messaging konnte größtenteils auf androsmex [1] zurückgegriffen werden, welches ebenfalls Spongycastle für die kryptographischen Verfahren und Algorithmen verwendet.

4.2 Hardwarekomponenten

Als Hardware wird ein Android-Gerät mit NFC-Schnittstelle vorausgesetzt. Sollen neben dem CHAT auch Informationen über das Zertifikat des Terminals angezeigt werden, muss der Chipsatz des Geräts und auch das Betriebssystem auf dem Smartphone extended length APDUs unterstützen, da die Zertifikate meist zu lang sind und sie in extended length APDUs an den nPA geschickt werden.

Zum Testen der Emulation fand das Android Smartphone Galaxy S III GT-I9300 in der 16 GB Version Verwendung. Das Smartphone lief mit der CyanogenMod-Version 9.1.0-i9300, die auf der Android Version 4.0.4 basiert. Der Cyanogenmod war erforderlich, um die komplette Kommunikation über NFC direkt an die App zu leiten, die die Smartcard emuliert. Die offizielle Host-based Card Emulation API (HCE), die ab Android Version 4.4 zur Verfügung steht, konnte für die Emulation des nPA nicht verwendet werden. HCE verlangt vom Terminal, dass es in der ersten APDU einen Application Identifier (AID) sendet, um eine Anwendung auszuwählen. Die zum Testen verwendete AusweisApp [2] sendet das entsprechende Kommando allerdings nicht. Somit ist eine Emulation gegenüber der AusweisApp nicht mit HCE möglich. Mit dem Cyanogenmod (Version 9) lässt sich die Notwendigkeit der Auswahl durch den AID umgehen.

Auf der anderen Seite wurden zum Testen die AusweisApp 1.13.0 unter Debian sowie unter Windows 8 und der ReinerSCT cyberJack RFID basis Contactless Smartcard Reader lokal verwendet. Bei der Authentisierung gegenüber dem eID-Service der Ausweis-Auskunft des Bürgerservice-Portals der Stadt Würzburg wurde die Emulation erfolgreich als nPA erkannt. Die Kommunikation brach erst ab, als die Zertifikate mit extended length APDUs verschickt wurden. Diese Einschränkung ist allerdings der Hardware des verwendeten Android-Geräts geschuldet.

4.3 Softwarekomponenten

Ziel der Implementierung ist es, die Software in möglichst unabhängige Komponenten zu unterteilen und somit eine klare Struktur, sowie Flexibilität bei der Weiterentwicklung zu erreichen.

Die App besteht aus folgenden Komponenten, die mit der Paketstruktur von Java gegliedert sind:

- **de.NPA.gui**
  Stellt die Klassen für die graphische Benutzeroberfläche bereit.
- **de.NPA.emulator**
  Enthält die Klassen, um den nPA zu emulieren sowie die Klassen zum Parsen des CHATs und der Zertifikatsbeschreibung.
- **de.NPA.nfc**
  Implementiert die NFC-Schnittstelle, die alle APDUs entgegen nimmt und versendet.
- **de.NPA.androsmex**
  Hier befinden sich die von androsmex übernommenen Komponenten [1]. Unter diesen ist die angepasste Implementation von PACE.

Die einzelnen Komponenten und die dazugehörigen Pakete sind in Abbildung 7 noch etwas detaillierter dargestellt. Hier ist auch zu erkennen, wie die einzelnen Komponenten miteinander interagieren: Die ankommenden APDUs werden vom "NFC-Service" an den

"NPA-Emulator" weitergereicht. Dieser verarbeitet die APDUs entweder selbst oder gibt sie weiter an die Komponenten "PACE" oder "TA". Sobald der "NPA-Emulator" den CHAT oder das Zertifikat erhält, kommt einer der beiden Parser (CHAT-Parser und CVCertificate-Parser) zum Einsatz. Die Informationen, die für den Benutzer relevant sind werden dann an die "GUI" (Graphische Benutzeroberfläche) weitergegeben und angezeigt.

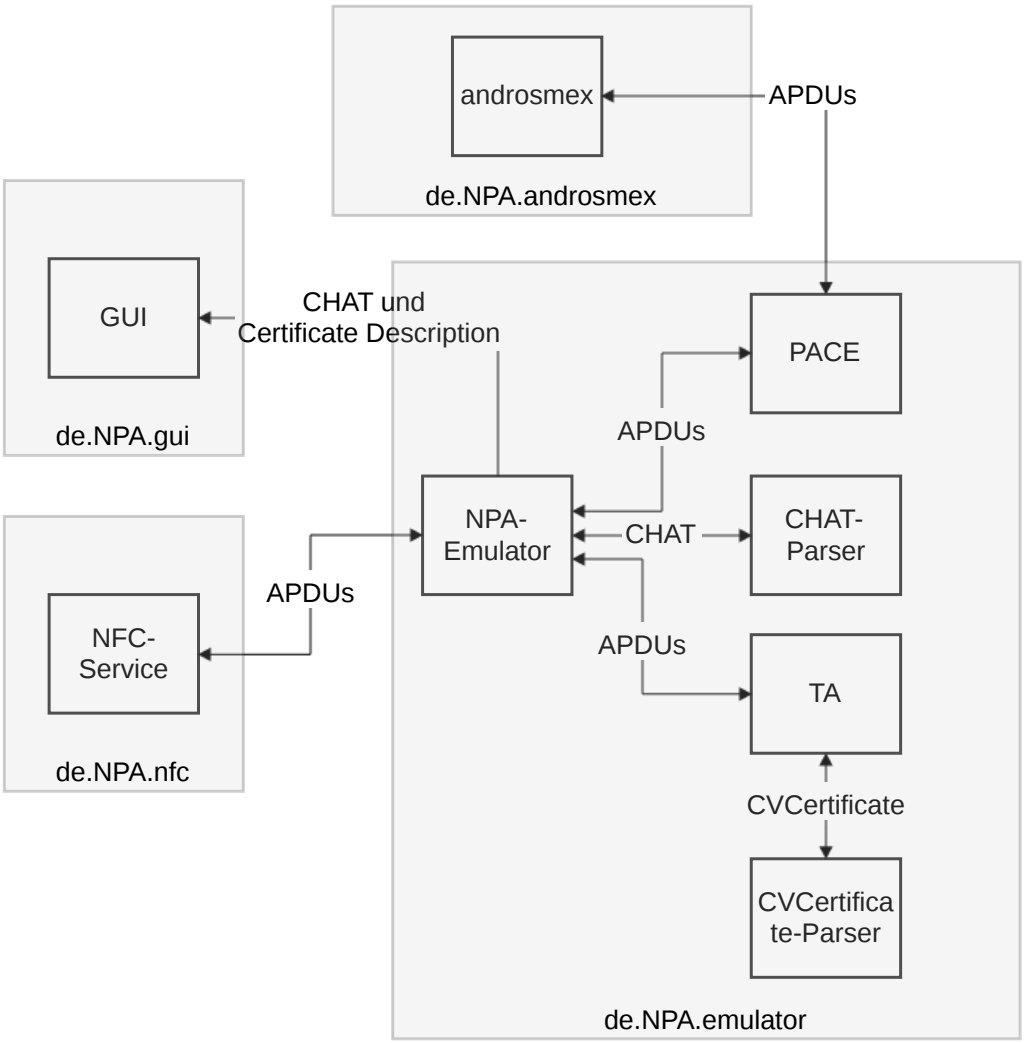

Abbildung 7: Detaillierte Darstellung der einzelnen Komponenten.

4.3.1 Graphische Benutzeroberfläche

Dem Nutzer wird eine graphische Benutzeroberfläche zur Verfügung gestellt. In einzelnen Kategorien kann der Nutzer die Informationen einsehen, die notwendig sind um ein

Terminal zu überprüfen. Zusätzlich gibt es noch die Möglichkeit, die einzelnen APDUs der Kommunikation zwischen dem Emulator und dem Terminal einzusehen.

Es wurde ein Tab-Layout verwendet, wodurch dem Nutzer eine leicht zu bedienende und übersichtliche Benutzeroberfläche geboten wird. Es lässt sich so leicht zwischen den Kategorien wechseln. Die einzelnen Kategorien sind folgende:

- **Access Rights:** Es werden die Zugriffsrechte des Terminals angezeigt, die dem CHAT entnommen wurden.
- **Certificate Information:** Hier werden Informationen zum Zertifikat angezeigt beziehungsweise zur Zertifikatskette angezeigt.
- **Log:** Zeigt die einzelnen APDUs der Kommunikation als Byte-Array an.

Die Informationen werden automatisch angezeigt, sobald sich das Handy mit einem entsprechenden Terminal verbindet.

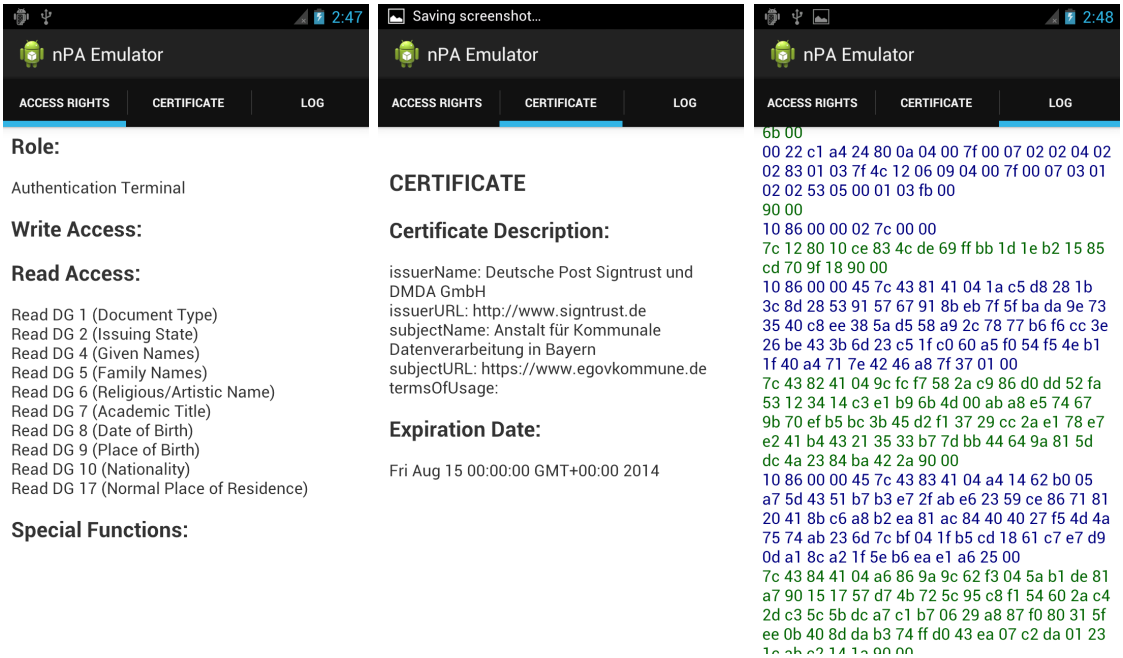

Abbildung 8: Screenshots der graphischen Benutzeroberfläche. Verwendet wurde die Ausweis-Auskunft des Bürgerservice-Portals der Stadt Würzburg zusammen mit der Ausweisapp. Die Daten für die Zertifikatsbeschreibung wurden der Anzeige der AusweisApp entnommen.

4.3.2 Emulator

Das Java-Paket Emulator ist das Kernstück der Anwendung und enthält folgende Klassen:

NPAEmulator: Die Klasse NPAEmulator verarbeitet alle eingehenden Command APDUs und koordiniert die anderen Teile des Emulators. Hier sind auch einige für die Emulation benötigten Kommandos gemäß ISO/IEC 7816 [8] implementiert. Dies sind zunächst die Kommandos Select File, Read Binary und Ask Random. Die Kommandos, die für PACE oder die TA implementiert sind befinden sich in den jeweiligen Klassen und nicht in der Klasse NPAEmulator.

PACE: Mithilfe der Klasse PACE wird PACE durchgeführt und ein Secure-Messaging-Kanal aufgebaut, der für alle nachfolgenden APDUs verwendet werden kann, um diese zu ver- und entpacken.Bei der Durchführung von PACE wird die angepasste Klasse PaceOperator aus androsmex verwendet. Als PIN kann prinzipiell jede beliebige PIN genutzt werden. In der Emulation wird die festgelegte PIN "123456" genutzt, die dann vom Benutzter auf dem PIN Pad des Terminals eingegeben werden muss. Für den operativen Einsatz sollte die PIN jedoch zufällig gewählt werden, damit das Terminal die Emulation nicht an dieser erkennt.

TA: Mit dieser Klasse werden Teile der TA durchgeführt. Die TA ist zu diesem Zeitpunkt nur bis zur Übertragung der Zertifikatskette implementiert [2]. So kann zwar eine Anzeige der Zertifikate erfolgen, aber keine Überprüfung, ob das Terminal den dazu gehörigen privaten Schlüssel besitzt. Spätestens bei der Authentisierung des Terminals gegenüber einem echten NPA würde die Authentisierung bei fehlendem privaten Schlüssel jedoch auffallen und die Authentisierung des Terminals verhindern.

Eine Erweiterung des Emulators durch die komplette Implementation der TA wäre notwendig, um schon bei der Überprüfung des Terminals durch den Emulator dessen Echtheit sicher feststellen zu können. Dadurch könnte verhindert werden, dass das Terminal bei Erkennen der Emulation eine fremde Identität vortäuscht und beim Durchlauf mit dem echten NPA die eigentliche Identität benutzt.

CVCertificateParser: Die Klasse CVCertifikateParser extrahiert aus einem Zertifikat die entsprechenden Informationen, die dem Nutzer angezeigt werden sollen.

CHATParser: Der CHATParser untersucht, welche Berechtigungen der CHAT enthält und gibt eine Beschreibung der einzelnen Berechtigungen an.

Mit diesen Klassen wird der NPA zum einen emuliert und zum anderen werden die entsprechenden Informationen extrahiert, die dem Nutzer später angezeigt werden sollen.

[2] Die vollständige Implementierung der TA war aufgrund des beschränkten Zeitrahmens und der eingeschränkten Hardware zum Testen nicht möglich.

4.3.3 NFC-Schnittstelle

Die NFC-Schnittstelle ist weitesgehend unabhängig von den anderen Komponenten und reicht lediglich als im Hintergrund laufender Service die eingehenden Command APDUs an die Klasse NPAEmulator weiter und nimmt von dieser die Response APDUs entgegen, um sie über die NFC-Schnittstelle an das Terminal zu übertragen.

Die ab Android 4.4 eingeführte Host-based Card Emulation (HCE) konnte aufgrund einer Einschränkung zunächst nicht verwendet werden. Die Einschränkung besteht darin, dass Android APDUs erst zum HCE Service einer App weiter leitet, sobald sich der NFC-Reader mit einem "SELECT AID" Kommando auf eine Application ID (AID) festgelegt hat. Da PACE allerdings im Master File (MF) ausgeführt werden soll, kommt es zum Problem.

CyanogenMod (in der Version 9.1.0) hat ein eigenes Interface für die host-based Card Emulation Unterstützung. Über die Interfaces IsoPcdA und IsoPcdB kann man das Android-Gerät mithilfe des Android-eigenen foreground dispatch Mechanismus in den Kartenemulations-Modus versetzen. Dadurch wird die komplette Kommunikation über NFC zur App weiter geleitet.

4.3.4 ASN.1

ASN.1 (Abstract Syntax Notation One) ist eine Beschreibungssprache zur Definition von Datenstrukturen. Die Datenobjekte, die während der Kommunikation mit dem nPA übertragen werden, sind BER-TLV-kodierte ASN.1-Datenobjekte und müssen entsprechend behandelt werden. BER (Basic Encoding Rules) beinhaltet dabei die Regeln, um die ASN.1-Datenobjekte platzsparend in einem Oktetstrom übertragen zu können. Bei der BER-TLV-Kodierung, wobei TLV für Tag-Length-Value steht, werden alle Datenobjekte mit einem Tag und einer Längenangabe versehen.

Um die benötigten Daten aus dem CHAT und aus den Zertifikaten zu extrahieren, müssen die BER-kodierten Datenobjekte entsprechend geparst werden, um sie danach für den Benutzer anzeigen zu können.

Für das verarbeiten der ASN1.Objekte wurden die Bibliotheken aus Spongycastle [11] verwendet. Spongycastle ist eine für Android angepasste Version der Sammlung kryptographischer Protokolle Bouncycastle [3]. Neben der Kompatibilität zu Android durch Spongycastle ist durch die Verwendung auch eine leichte Integration von androsmex gewährleistet, da auch hier Spongycastle verwendet wird.

4.3.5 androsmex

Androsmex [1] implementiert das PACE-Protokoll für NFC-fähige Android-Smartphones und führt PACE in der frühen Alpha-Version mit dem nPA aus. Da PACE in androsmex allerdings nur für die Seite des Terminals implementiert ist, waren einige Anpassungen und Erweiterungen notwendig, um die Implementation für die Emulation des nPA zu nutzen.

Nach dem Durchlauf von PACE wird von androsmex ein SecureMessaging-Objekt erstellt, mit dem eingehende und ausgehende APDUs mit den zuvor ausgehandelten

Schlüsseln ver- und entpackt werden können.

In der Klasse SecureMessaging in androsmex mussten die Funktionen zum Verpacken von Response APDUs und Entpacken von Command APDUs implementiert werden, da für die Implementierung von PACE für das Terminal nur die komplementären Funktionen - also zum Verpacken von Command APDUs und Entpacken von Response APDUs - benötigt werden.

5 Ausblick

Die in dieser Arbeit vorgestellte Lösung zeigt, dass es prinzipiell möglich ist, Teile des nPA zur Überprüfung eines Terminals zu emulieren, auch wenn es dabei noch Schwierigkeiten gibt. Mit einer entsprechenden Unterstützung der extended length APDUs im Kartenemulations-Modus durch die Hardware und ohne die Einschränkung des Android-Betriebssystems, die in Kapitel 4.3.3 aufgezeigt ist, ließe sich die Prüfung unkompliziert bewerkstelligen. Aufgrund dieser Schwierigkeiten ist es momentan nicht möglich, die Implementierung auf jedem beliebigen Android-Gerät mit NFC-Unterstützung zu verwenden. Es muss zunächst noch auf die Cyanogenmod Version 9 zurückgegriffen werden, die nicht für alle Android-Geräte zur Verfügung steht. Abgesehen von der Behebung dieser Probleme könnte in einem nächsten Schritt die TA vollständig implementiert werden. Dadurch wäre gewährleistet, dass das Zertifikat des zu überprüfenden Terminals echt ist. Auch eine Datenbank, die die jeweiligen Hashwerte auf die Zertifikatsbeschreibung abbildet, müsste noch erstellt werden.

Durch die breits jetzt große Verbreitung von NFC-fähigen Android-Geräten bietet die Implementierung für Android einen Weg der Prüfung, der für viele nicht mit keinen Anschaffungskosten der Hardware verbunden ist. Prinzipiell ist es also möglich, ein Terminal auch von Menschen ohne große technische Kenntnisse auf einfache Weise zu prüfen.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, den 24. September 2014 .....

Literatur

[1] *androsmex - a PACE implementation for android smartphones with NFC capabilities.* https://code.google.com/p/androsmex/. – Online; Seite aufgerufen am 22.09.2014

[2] *Die AusweisApp.* https://www.ausweisapp.bund.de. – Online; Seite aufgerufen am 22.09.2014

[3] *The Legion of the Bouncy Castle.* https://www.bouncycastle.org/. – Online; Seite aufgerufen am 22.09.2014

[4] Bundesamt für Sicherheit in der Informationstechnik: *Architektur elektronischer Personalausweis und elektronischer Aufenthaltstitel.* https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03127/tr-03127.html. – Online; Seite aufgerufen am 22.09.2014

[5] Bundesamt für Sicherheit in der Informationstechnik: *BSI TR-03110: Advanced Security Mechanisms for Machine Readable Travel Documents - Part 1 – Common Specifications.* 2013

[6] Horsch, Moritz: *MobilePACE - Password Authenticated Connection Establishment implementation on mobile devices*, Technische Universität Darmstadt, Bachelorarbeit, 2009. https://www.cdc.informatik.tu-darmstadt.de/mona/pubs/200909_BA_MobilePACE.pdf. – Online; Seite aufgerufen am 22.09.2014

[7] *ISO 14443: Identification cards – Contactless integrated circuit(s) cards – Proximity cards.* 2000

[8] *ISO/IEC 7816-1:1998, Identification cards - Integrated circuit(s) cards with contacts - Part 1: Physical characteristics.* 2007

[9] Morgner, Frank: *Mobiler Chipkartenleser für den neuen Personalausweis*, Humboldt-Universtität zu Berlin, Diplomarbeit, 2012. http://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2012-05/SAR-PR-2012-05_.pdf. – Online; Seite aufgerufen am 22.09.2014

[10] Oepen, Dominik: *Authentisierung im mobilen Web - Zur Usability eID basierter Authentisierung auf einem NFC Handy*, Humboldt-Universtität zu Berlin, Diplomarbeit, 2010. https://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2010-11/SAR-PR-2010-11_.pdf. – Online; Seite aufgerufen am 22.09.2014

[11] *Spongy Castle - repackage of Bouncy Castle for Android.* http://rtyley.github.io/spongycastle/. – Online; Seite aufgerufen am 22.09.2014

[12] *BSI TR-03110: Advanced Security Mechanisms for Machine Readable Travel Documents - Part 3 – Common Specifications.* 2013

[13] *EAC-PKI'n für den elektronischen Personalausweis*. https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03128/index_htm.html. – Online; Seite aufgerufen am 22.09.2014