Sichere Bereitstellung von Identitätstoken auf mobilen Endgeräten

Diplomarbeit

zur Erlangung des akademischen Grades
Diplominformatiker

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

eingereicht von: Martin Schröder

Gutachter(innen): Prof. Dr. Jens-Peter Redlich
Dr. Manfred Paeschke

eingereicht am: .....

verteidigt am: .....

Zusammenfassung

Durch die zunehmende Vernetzung der Gesellschaft gewinnt die mobile Internetnutzung und damit der Wunsch, sich sicher mobil authentisieren zu können, immer mehr an Bedeutung. Trotzdem ist die Authentisierung per Benutzername und Passwort immernoch weit verbreitet. Im stationären Bereich hat sich Authentifizierung per Chipkarte bereits als eine sichere Alternative etabliert. Die Handhabung mit einer Chipkarte erscheint jedoch im mobilen Einsatz als zu umständlich, unter anderem weil dadurch der Sinn eines mobilen Endgerätes, nämlich sämtliche Aufgaben mobil mit einem Gerät erledigen zu können, wiederum in Frage gestellt wird. Eine mögliche Alternative bieten Identitätstoken, bei denen es sich um speziell signierte Daten handelt, die zur Authentisierung verwendet werden können.

Die Diplomarbeit beschäftigt sich mit zwei Problemen. Zum einen wird die sichere Speicherung der Identitätstoken im Endgerät betrachtet. Mit einem speziell gesicherten Chip im Endgerät, einem Secure Element, werden dabei unerlaubte Zugriffe auf bestimmte Teile des Tokens verhindert. Für die Verwendung eines Identitätstoken ist somit das Secure Element zwingen erforderlich, welches dafür mit einer PIN freigeschaltet werden muss.

Das zweite Problem stellt die Ausstellung des Identitätstokens dar. Da das Secure Element im Auslieferungszustand noch keinem Benutzer zugeordnet ist, ist bei der Ausstellung nicht sichergestellt, dass der Token in dem korrekten Gerät gespeichert wird. Aus dem Grund wird in der Arbeit ein zweistufiges Personalisierungsverfahren vorgestellt, bei dem das Secure Element mit Benutzerattributen personalisiert wird. Dieses Verfahren kann auch in einer unsicheren Umgebung beim Benutzer durchgeführt werden. Die anschließende Tokenausstellung baut dann auf der Personalisierung auf.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACS	Anonymous Credential System
BSI	Bundesamt für Sicherheit in der Informationstechnik
CA	Chip Authentication
CC	Common Criteria
CIN	Card Image Number
DAP	Data Authentication Pattern
DM	Delegated Management
EAC	Extended Access Control
EAL	Evaluation Assurance Level
IIN	Issuer Identification Number
ISD	Issuer Security Domain
JC	Java Card
NFC	Near Field Communication
nPA	Neuer Personalausweis
PACE	Password Authenticated Connection Establishment
PKI	Public Key Infrastructure
PSK	Pre-Shared Key
REE	Rich Execution Environment
rID	Restricted Identifier
SD	Security Domain
SE	Secure Element
SIM	Subscriber Identity Module
SSD	Supplementary Security Domain
TA	Terminal Authentication
TEE	Trusted Execution Environment
TSM	Trusted Service Manager
UICC	Universal Integrated Circuit Card
UMTS	Universal Mobile Telecommunications System
WLAN	Wireless Local Area Network

1. Einleitung

1.1. Einführung

Mobile Endgeräte, gemeint sind hier Mobiltelefone, Smartphones und Tablet-PCs, haben in letzter Zeit immer mehr an Bedeutung gewonnen. Funktionierte bis vor einigen Jahren das mobile Surfen im Internet nur auf wenigen speziellen Internetseiten, kann man sich mittlerweile mit Smartphones fast so komfortabel im Internet bewegen wie mit stationären Rechnern. Unter anderem deswegen hat sich allein innerhalb der letzten zwei Jahre der Anteil der mobilen Internetnutzung von 11% auf 20% annähernd verdoppelt [12]. Gleichzeitig rücken die mobilen Endgeräte nicht zuletzt durch die weite Verbreitung und die gestiegene Softwarekomplexität immer mehr in das Visier von Angreifern.

Aus diesem Grund werden in immer mehr mobilen Endgeräten spezielle Sicherheitschips, sogenannte Secure Elements, verbaut. Diese sind unabhängig von dem unsicheren Betriebssystem des Gerätes. Eine Anwendung, die für die Nutzung des Secure Elements prädestiniert ist, stellt die elektronische Brieftasche dar, die Kreditkartendaten sicher speichern kann. Das Konzept der elektronischen oder auch virtuellen Brieftasche wurde bisher hauptsächlich für das elektronische Bezahlen umgesetzt.

Eine konsequente Weiterführung besteht nun darin, auch virtuelle Ausweisdokumente mit in die Brieftasche aufzunehmen und so eine elektronische Identitätsfunktion (eID-Funktion) zu realisieren. Mit dieser Funktion kann sich der Benutzer online sowie offline ausweisen. Technisch wird solch ein virtuelles Dokument durch einen oder mehrere sogenannter Identitätstoken realisiert. Dabei handelt es sich um ein Konstrukt aus signierten Daten, das über spezielle datenschutzfreundliche Eigenschaften verfügt. Die Bereitstellung eines Identitätstokens ist eine große Herausforderung. Um einem Token eine möglichst große Beweiskraft beizumessen, muss er nachweislich von einer authentischen Datenquelle abgeleitet und vor unbefugtem Auslesen und Verändern geschützt gespeichert werden. Gleichzeitig muss bei der Ausstellung sichergestellt werden, dass der Identitätstoken in das korrekte Gerät geschrieben wird.

1.2. Zielstellung

Das Ziel der Diplomarbeit ist es, ein Verfahren zu entwerfen, mit dem ein von einer authentischen Datenquelle abgeleiteter Identitätstoken auf einem mobilen Endgerät bereitgestellt werden kann. Mit Hilfe des Tokens kann sich der Besitzer des Endgerätes später mobil

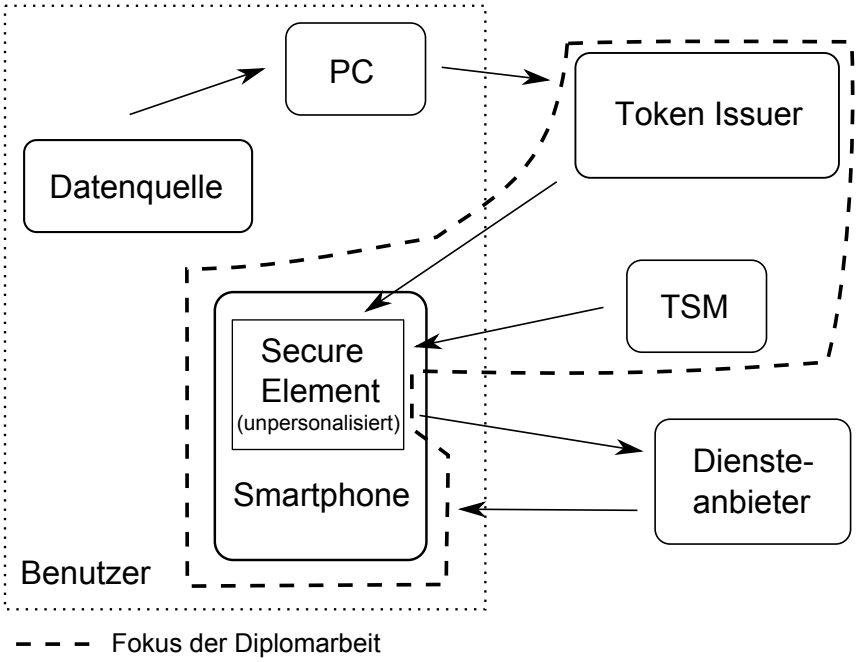

Abbildung 1.1.: Übersicht

authentisieren. Die Abbildung 1.1 stellt einen Überblick dar, wobei sich die Diplomarbeit auf die Ersteinrichtung des Secure Elements (SE) sowie die sichere Übertragung und Speicherung des Tokens konzentriert. Für die Einrichtung des SE ist ein vertrauenswürdiger Dienst, der Trusted Service Manager (TSM), zuständig. Nachdem das Secure Element initialisiert wurde, werden die Attribute mit Hilfe des PCs aus der authentischen Datenquelle ausgelesen und an den Token Issuer gesendet. Bei dem Token Issuer handelt es sich um einen Dienst, der aus den Attributen Identitätstoken ableitet und sie an das mobile Endgerät sendet. Letztendlich müssen mehrere Probleme gelöst werden:

I. Bei der Übertragung muss die Vertraulichkeit und Integrität der Identitätstoken gewährleistet sein.

   Auch wenn bereits Techniken zur sicheren Datenübertragung bestehen, sind diese eventuell nicht direkt auf das Szenario anwendbar. Mögliche Probleme können sich durch die Einbettung des Secure Elements in das Smartphone ergeben, wodurch Angriffe durch Schadsoftware auf dem Smartphone ermöglicht werden. Des Weiteren sollte betrachtet werden, inwiefern Secure Element und Token Issuer bereits Kenntnis voneinander haben müssen, um Man-in-the-middle Angriffe verhindern zu können.

II. Die Identitätstoken müssen in dem korrekten Gerät gespeichert werden.

   Hierbei sollen Angriffe abgewehrt werden, bei denen die Daten des Benutzers in ein vom Angreifer kontrolliertes Gerät umgeleitet werden. Dies ist relativ leicht sicherzustellen, wenn der Token Issuer und das Secure Element direkt miteinander verbunden

sind. Erfolgt die Kommunikation allerdings über weitere Zwischenstationen (Internetdienstanbieter, Smartphone,..), müssen zusätzliche Vorkehrungen getroffen werden. Eine zusätzliche Herausforderung ist es, wenn der Token Issuer keine Zuordnung zwischen Benutzer und SE besitzt. Dabei könnte er im Extremfall nur sicherstellen, dass die Daten an ein authentisches SE übertragen werden, allerdings nicht, ob es dem Benutzer gehört.

III. Es dürfen nur berechtigte Akteure in vorgegebener Weise auf den Datenspeicher zugreifen.

   Neben Schadprogrammen und unberechtigten Dritten kann auch der Benutzer oder der Token Issuer nicht befugt sein, bestimmte Operationen auf den Daten auszuführen. So könnte dem Token Issuer der Einblick in die Daten verwehrt werden, um die Vertrauenswürdigkeit des Systems zu erhöhen oder dem Benutzer das Verändern und Kopieren der Daten verweigert werden, damit deren Authentizität gewahrt bleibt.

IV. Das System sollte konform zu bestehenden Standards sein.

   Auch wenn es ein Ziel der Diplomarbeit ist, einen eigenen Forschungsbeitrag zu leisten, sollte dieser auf bereits bestehenden Standards aufbauen. Diese haben sich bereits etabliert und wurden in der Regel Sicherheitsbetrachtungen unterzogen, wodurch sich Akzeptanz und Sicherheit des Systems erhöht.

V. Das System sollte performant und leicht nutzbar sein.

   Auch wenn der Fokus auf die Sicherheit der Benutzerdaten gelegt wird, spielt ebenfalls die Leistung und Nutzbarkeit des Systems eine Rolle. Schließlich stellen diese einen Indikator für die Komplexität und somit für die Akzeptanz der Lösung dar.

1.3. Verwandte Arbeiten

Für die Arbeit spielen die Bereiche Sicherheit, Datenschutz und Identitätsmanagement eine bedeutende Rolle. Da die Forschungsergebnisse in diesen Bereichen sehr zahlreich sind, werden an dieser Stelle nur ausgewählte Veröffentlichungen genannt, in denen unterschiedliche Schwerpunkte gesetzt wurden. Auf Veröffentlichungen in den beiden großen Bereichen *Anonymous Credential Systems* [39, 11] und *Chipkartenverwaltung* [3] wird in den Kapiteln 2.3.3, 2.2 und 2.2.3 eingegangen.

In seiner Veröffentlichung mit dem Titel „An Open Mobile Identity Tool: An Architecture for Mobile Identity Management“ [33] beschreibt Konstantin Hyppönen unter anderem eine Lösung zur mobilen Authentifizierung, die auf dem finnischen eID-System namens FINeID aufsetzt. FINeID basiert auf zwei Zertifikaten, ein Authentisierungs- und ein Signaturzertifkat, die mit den zugehörigen privaten Schlüsseln im Personalausweis oder einer speziellen SIM-Karte (PKI-SIM) liegen. Hyppönen setzt voraus, dass das Zertifikat als

einziges Datum einen Hash über ein biometrisches Foto[1] enthält und zusammen mit den Benutzerdaten und dem Foto in der PKI-SIM gespeichert ist. Für die mobile Authentifizierung wählt ein spezielles Applet[2] auf der SIM-Karte nur die benötigten Attribute aus, signiert diese und schickt sie zusammen mit dem Zertifikat und dem biometrischen Foto zur Gegenstelle. Diese überprüft die Signatur und kann durch eine Sichtprüfung sicherstellen, dass das Zertifikat dem Benutzer gehört. Nachteil des Systems ist, dass der Benutzer bei jeder Authentifizierung das biometrische Merkmal preisgibt. Dies bedeutet nicht nur, dass überflüssige Daten übertragen werden, sondern auch, dass der Benutzer darüber verfolgbar ist und somit nicht mehr vollständig anonym handeln kann. Des Weiteren funktioniert das System nur bei persönlichem Kontakt, bei dem eine Überprüfung des biometrischen Merkmales sichergestellt werden kann.

Lasse Edlund beschäftigt sich in seiner Master Thesis mit dem Titel „Secure and Confidential Applications on UICC“ [30] mit der Frage, wie mehrere Applikationen sicher in einem UICC (Universal Integrated Circuit Card) verwaltet werden können und untersucht zusätzlich, ob NFC-fähige Smartphones bereits vorhandene RFID-Karten ersetzen könnten. Dabei legt er großen Wert darauf, seine Lösung zu bestehenden Standards, wie denen von ETSI oder GlobalPlatform kompatibel zu gestalten. Bei dem von ihm vorgestellen Prototypen teilt er das UICC in verschiedene Security Domains auf, wie es in der GlobalPlatform Card Specification [18] beschrieben wird. Diese stellen spezielle Bereiche dar, auf die nur die entsprechenden Besitzer der Security Domain zugreifen können. Um ein Applet in das UICC zu spielen, wird es über Kurznachrichten von einem speziellen Server an dieses gesendet. Dort gelangt es zuerst in den Bereich des UICC-Herstellers, der Issuer Security Domain, und kann von dort aus in die Security Domain des Applet Herstellers verschoben werden. Der Nachteil an dem Vorgehen ist, dass eine starke Zusammenarbeit mit den jeweiligen Mobilfunkanbietern erforderlich ist, da diese die Hoheit über ihre SIM-Karten besitzen. Somit benötigt man ihre Unterstützung, um zum einen die Daten über Short Messages versenden zu können und zum anderen, um eine eigene Security Domain zu bekommen, in welche das Applet dann gespielt werden kann. Auch wenn einige Mobilfunkanbieter diesen Service mittlerweile anbieten, hat ein Applethersteller trotzdem das Problem, dass er mit allen benötigten Mobilfunkanbietern Verträge eingehen muss.

Einen offeneren Ansatz zur mobilen Authentifizierung stellt Christian Hansen in seiner Master Thesis mit dem Titel „A Framewok for Identity and Privacy Management on Mobile Devices“ [32] vor. Nach ihm haben bisherige Lösungen den Nachteil, dass sie entweder wegen statischer Pseudonyme bzw. Zertifikaten nicht anonym genug oder sie auf Grund der Integration in die SIM-Karte, wie es zum Beispiel in [33] der Fall ist, zu teuer für die Anbieter sind. Seine Lösung soll unter anderem diese Probleme umgehen, indem die Daten verschlüsselt im Smartphonespeicher abgelegt werden, wobei der verwendete symmetrische Verschlüsselungsschlüssel verschlüsselt auf einem extra Server liegt. Die Authentifizierungsanfragen werden von einem speziellen Client auf dem Smartphone bearbeitet, der sich alle bisherigen Abfragen merkt und den Benutzer darauf hinweist, welche Daten er bereits

[1] Prinzipiell sind auch andere biometrische Merkmale, wie zum Beispiel ein Fingerabdruck, denkbar.
[2] Damit ist eine Anwendung gemeint, die auf der SIM-Karte ausgeführt wird.

an den Diensteanbieter übergeben hat, um ihn so vor einer möglicher Zusammenführung seiner Daten warnen zu können. Zusätzlich kann der Client ein dynamisches Pseudonym pro Anbieter generieren. Allerdings lässt Hansen offen, aus welcher Datenquelle die Benutzerattribute kommen und somit auch wie die Authentizität dieser sichergestellt werden kann. Auch belastet die von ihm vorgestellte anonymisierende Netzwerkinfrastruktur, die Ähnlichkeiten mit Tor [6] aufweist, den Akku und die Internetverbindung der mobilen Endgeräte.

Das seit November 2010 laufende Projekt ABC4Trust[3] hat sich als Ziel gesetzt, die Verbreitung von Attribut basierten Credential Systemen zu fördern und der Allgemeinheit zugänglich zu machen. Bei diesen Systemen bekommt ein Nutzer von einem Identity Provider eine Art signiertes Zertifikat ausgestellt, das Attribute über ihn enthält. Zur Authentifizierung bei einem Dienst kann er nun dieses Zertifikat verwenden, wobei er selber bestimmen kann, welche Attribute er dem Dienst preisgibt. Leider sind bestehende Systeme zur Zeit relativ komplex und finden deswegen kaum Verbreitung. Das Ziel des bis Oktober 2014 laufenden Projekts ist es, diese Systeme zu kapseln, sowie ihre Benutzung für den Anwender zu erleichtern.

Die ersten beiden Arbeiten haben gemein, dass sie eigene Applets auf der SIM-Karte benötigen. Zusätzliche Hardwaresicherheit ist zwar bei der Speicherung und Verwendung von sensiblen Daten von Vorteil, jedoch erfordern diese Lösungen die Zusammenarbeit mit den Mobilfunkanbietern. Fällt, wie bei der dritten Arbeit, der Hardwareschutz weg, können die Daten kopiert oder durch Offlineattacken leichter analysiert werden. Aus diesem Grund bietet sich die Verwendung eines Secure Elements an, was jedoch in den genannten Arbeiten nicht betrachtet wurde und dem bislang allgemein in der Literatur wenig Bedeutung zugekommen ist. Eine Ausnahme bildet das ABC4Trust Projekt, bei dem unter anderem das Zusammenspiel von Anonymous Credential Systems und Secure Elements betrachtet wird. Das Projekt ist allerdings noch nicht abgeschlossen.

1.4. Eigener Beitrag

In der vorliegenden Arbeit wird ein Verfahren vorgestellt, mit dem ein Identitätstoken sicher auf einem mobilen Endgerät, das über ein Secure Element verfügt, bereitgestellt werden kann. Dafür werden verschiedene bestehende Protokolle, Spezifikationen und Standards kombiniert.

Grundlage ist die Analyse verschiedener Speichermöglichkeiten von Identitätstoken in mobilen Endgeräten, insbesondere der geteilten Speicherung im Dateisystem und Secure Element. Besonderer Wert wird dabei auf die Verarbeitungsgeschwindigkeit sowie die Sicherheit der Token gelegt.

Die Einrichtung eines Secure Elements wird betrachtet. Es wird gezeigt, wie eine Security Domain angelegt und danach das entsprechende Applet installiert werden kann. In der

[3] `https://abc4trust.eu`

anschließenden Sicherheitsbetrachtung wird aufgezeigt, dass diese Verfahren anfällig für Weiterleitungsangriffe sind, was eine Personalisierung des Secure Elements nötig macht.

Anschließend wird ein Verfahren vorgestellt, mit dem sich das SE in einer unsicheren Umgebung beim Benutzer personalisieren lässt und das robust gegen die meisten Weiterleitungsangriffe ist. Die Personalisierung baut auf einem zweistufigen Verfahren auf, bei dem sich an die eigentliche Personalisierung eine Verifikationsphase anschließt. Darauf folgt die Ausstellung der Identitätstoken, welche auf der erfolgreichen Personalisierung aufbaut.

1.5. Aufbau

Kapitel 2 stellt die Grundlagen für die Arbeit bereit und den Ist-Stand bei mobilen Endgeräten, Secure Elements und der Authentifizierung im Internet vor. Die Erkenntnisse der Grundlagen münden in Kapitel 3 „Anforderungen“. Dort wird dargestellt, welcher Schutzbedarf besteht, was für Annahmen getroffen werden können und über welche Möglichkeiten ein Angreifer verfügt. Anhand der Anforderungen wird in Kapitel 4 eine Lösung zur Speicherung und in Kapitel 5 zur Bereitstellung der Identitätstoken präsentiert, die jeweils unter Sicherheitsaspekten betrachtet werden. In Kapitel 6 erfolgt die Auswertung der Arbeit, indem die vorgeschlagene Lösung evaluiert wird und Schlussfolgerungen gezogen werden. In einem kurzen Ausblick wird zum Schluss diskutiert, welche Verbesserungen denkbar sind und in welchen Themengebieten die Arbeit zusätzlich von Bedeutung sein könnte.

2. Grundlagen und Stand der Technik

2.1. Mobile Endgeräte

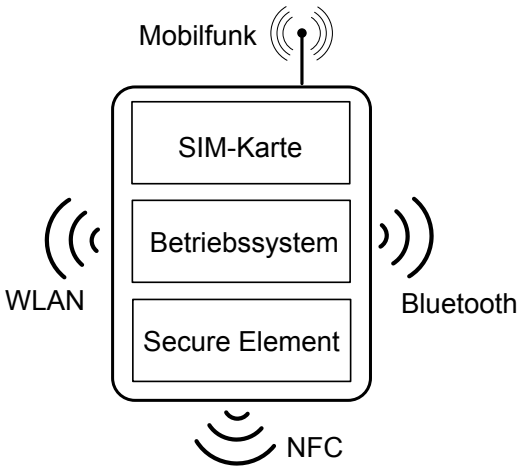

Abbildung 2.1.: Überblick Endgerät

Dank der einfachen Entwicklung mobiler Applikationen und der weiten Verbreitung mobiler Endgeräte gibt es mittlerweile eine Reihe an Schadprogrammen dafür. In Abschnitt 2.1.2 wird deswegen betrachtet, welche Sicherheitskonzepte in einem aktuellen mobilen Betriebssystem existieren. Da Smartphones neben der Mobilfunkschnittstelle zahlreiche weitere Schnittstellen wie USB, WiFi, Bluetooth oder NFC besitzen, die sich prinzipiell auch zur Übertragung von Identitätstoken eignen, wird auf diese in Sektion 2.1.1 näher eingegangen. Ebenfalls wichtig sind Secure Elements, die in immer mehr Smartphones verbaut werden oder als microSD-Karte nachgerüstet werden können. Sie stellen, wie die SIM-Karte, eine sichere Ausführungsumgebung zur Verfügung und sind somit zur Speicherung der Identitätstoken interessant, weswegen sie in dem Kapitel 2.2 behandelt werden.

2.1.1. Drahtlose Schnittstellen

In diesem Abschnitt wird betrachtet, inwiefern sich die Schnittstellen von mobilen Endgeräten zur Übertragung der Identitätstoken eignen. Die Schnittstellen besitzen sehr unterschiedliche Eigenschaften. Für die Arbeit sind vor allem die implementierten Sicherheitsmechanismen sowie der Verbreitungsgrad wichtig. Ebenfalls kann die Reichweite eine große

Rolle spielen. Die hier vorgestellten Schnittstellen werden bis auf NFC in fast allen aktuellen mobilen Endgeräten verbaut. NFC selber konnte sich in seinem 10-jährigen Bestehen noch nicht wirklich im mobilen Bereich durchsetzen, soll hier aber auf Grund des guten Zusammenspiels mit Secure Elements ebenfalls betrachtet werden.

Mobilfunk

Seit den Anfangszeiten des Mobilfunks haben sich die Datenrate und die Anzahl der zur Verfügung stehenden Dienste stark vergrößert. Den ersten weltweit einheitlichen Mobilfunkstandard stellt Global System for Mobile Communications (GSM) dar. Die Übertragung zwischen Endgerät und Sendemast erfolgt bei GSM digital und verschlüsselt, wobei für die Verschlüsselung ein proprietärer zunächst geheimer Algorithmus benutzt wird, dessen Unsicherheit mittlerweile bewiesen ist und aktiv ausgenutzt wird [38]. Für die Authentifizierung des Benutzers ist mit dem Subscriber Identity Module (SIM) eine spezielle Karte zuständig, die in das Telefon eingesteckt wird. Bei dem nachfolgenden Mobilfunkstandard, in Deutschland unter UMTS bekannt, wurde die SIM-Karte in Universal Integrated Circuit Card (UICC) umbenannt. Die UICC stellt neben ursprünglichen Funktionen der SIM auch eine sichere Ausführungsumgebung für sensible Anwendungen zur Verfügung. Die Verschlüsselungsalgorithmen der „letzten Meile“ wurden mit der Zeit immer weiter verbessert. Trotzdem können die Mobilfunkanbieter prinzipiell alle Gespräche mithören. Eine Ende-zu-Ende-Verschlüsselung lässt sich zwar per Internettelefonie einrichten, jedoch muss hierfür meistens Zusatzsoftware auf dem Endgerät installiert werden.

WLAN

Wireless Local Area Network (WLAN) steht für eine Reihe von Standards für drahtlose lokale Netzwerke, wobei hier insbesondere die IEEE 802.11 Familie gemeint ist. Der ursprüngliche Standard wurde 1997 verabschiedet und versprach eine maximale Bruttodatenrate von 2 MBit/s im lizenzfreien 2,4-GHz-Band. Dieser wurde in den folgenden Jahren stetig ergänzt, wobei unter anderem die heute vielfach genutzten Erweiterungen 802.11b, g und n entstanden sind. Neben der Erhöhung der Datenrate wurde auch das Sicherheitsprotokoll verbessert. Das mit 802.11b eingeführte Wired Equivalent Privacy (WEP) ist für die Autorisierung der WLAN-Teilnehmer und die Integrität und Vertraulichkeit der übermittelten Daten zuständig. Als Standard-Authentifizierungsmethode wird Open System Authentication benutzt, bei der der WEP-Schlüssel gleichzeitig der Authentifizierung dient. Der RC4-Algorithmus dient der Verschlüsselung, die Datenintegrität wird mit CRC32 sichergestellt. Relativ bald nach dessen Veröffentlichung wurden verschiedene Schwachstellen in WEP bekannt, sodass innerhalb kurzer Zeit ein Nachfolger benötigt wurde. Da der neue Sicherheitsstandard IEEE 802.11i noch nicht fertig spezifiziert war, entschloss man sich, Teile davon in Form von Wi-Fi Protected Access (WPA) als Übergangslösung zu verwenden. Nachdem der neue Standard dann 2004 verabschiedet wurde,

konnte er in den ersten Geräten in Form des heute noch als sicher geltenden WPA2 implementiert werden. Hierbei wird zur Verschlüsselung AES eingesetzt, die Authentifizierung erfolgt meistens über einen Pre-Shared Key (PSK), der auf beiden Seiten bekannt sein muss. Ein zu kurz gewählter PSK kann allerdings das System anfällig für Bruteforceattacken werden lassen. Alles in allem liegt der Vorteil von WLAN in der hohen Reichweite von bis zu 100 Meter bei einer ebenfalls hohen Datenrate. Jedoch ist das System störanfällig, sodass es zu Verbindungsabbrüchen kommen kann, wenn in der Nähe weitere WLAN-Netze oder andere Geräte, wie Mikrowellen oder Bluetooth, die auf dem selben Band arbeiten, betrieben werden.

Bluetooth

Bluetooth[1] ist ein Funkstandard, um Geräte über mehrere Meter Entfernung zu verbinden und somit Kabelverbindungen zu ersetzen. Hauptsächlich wird er für die Kommunikation zwischen Computern und Peripheriegeräten verwendet. Bluetooth verwendet wie WLAN das lizenzfreie 2,4-GHz-Band, ist jedoch dank des Frequenzsprungverfahrens robuster gegen Störungen[2]. Durch das verwendete Verfahren ist die Übertragungsrate mit 2,1 Mbit/s jedoch vergleichsweise gering. Die neue Bluetooth low energy Erweiterung[3] ist mit 1 Mbit/s sogar noch langsamer und hat nur eine Reichweite von 10 Metern, erlaubt dafür jedoch einen Verbindungsaufbau innerhalb von nur fünf Millisekunden. Um die Authentizität der Kommunikationspartner und Vertraulichkeit der Daten sicherzustellen, kann wie bei WLAN ein PSK eingegeben werden. Dafür müssen die Geräte jedoch trivialerweise über eine Eingabemöglichkeit verfügen. Peripherie wie Freisprecheinrichtungen oder Computermäuse verwenden meist einen Standardschlüssel, der somit die Kommunikation unsicher werden lässt. Alternativ kann eine Verbindung auch über NFC initialisiert werden. Zusammenfassend lässt sich sagen, dass Bluetooth zwar keine so hohe Datenrate wie die aktuellen WLAN-Standards bietet, dafür jedoch dank seiner Reichweite und dem raschen Verbindungsaufbau eine komfortablere, aber auch unsichere Verbindung zwischen zwei Geräten bietet, als es per Kabel oder NFC der Fall ist.

NFC

Near Field Communication (NFC) ist eine 2002 von den Firmen Sony und Philips entwickelte Funktechnik über sehr kurze Distanz, für dessen Weiterentwicklung seit 2004 das NFC-Forum[4] zuständig ist. Die grundlegenden Eigenschaften, wie die verwendete Frequenz von 13,56 MHz, die Reichweite von bis zu 20 Zentimetern und die maximale Datenrate von 424 kBit/s wurden dabei bewusst so gewählt, dass NFC-Geräte auch mit ISO/IEC 14443

[1] `http://www.bluetooth.com/Pages/what-is-bluetooth-technology.aspx`

[2] Dabei wird das Frequenzband in einzelne Teilbänder aufgeteilt, die mehrere hundert Mal in der Sekunde gewechselt werden. Werden einzelne Teilbänder gestört, gehen dadurch nur wenige aufeinander folgende Pakete verloren, was durch die Fehlerkorrektur ausgeglichen werden kann.

[3] `http://www.bluetooth.com/Pages/low-energy.aspx`

[4] `http://www.nfc-forum.org/home/`

konformen Chipkarten kommunizieren können. Darüber hinaus stellt es eine Verbindungstechnologie dar, mit der zwei aktive NFC-fähige Geräte miteinander kommunizieren können, was für eine Abkehr von dem starren Master-Slave-Prinzip in der Chipkartenkommunikation steht. Genau genommen kann ein NFC-Gerät in drei verschiedenen Betriebsarten verwendet werden:

Aktiver Modus Beide Kommunikationspartner generieren ein Hochfrequenzfeld mit der Trägerfrequenz zum Senden von Daten. Dieser Modus wird auch Peer-to-Peer Modus genannt und dient dem Datenaustausch zwischen zwei Endgeräten. So können zum Beispiel Informationen zum Aufbau einer Bluetoothverbindung ausgetauscht werden. Es existieren auch Kartenleser, die diesen Modus unterstützen.

Passiver Modus als Master Nur das Gerät generiert ein Hochfrequenzfeld mit der Trägerfrequenz, während der Kommunikationspartner die Daten per Lastmodulation überträgt. Dies entspricht dem Master-Slave-Prinzip von ISO/IEC 14443 und deckt somit alle Anwendungsfälle ab, in denen ein NFC-fähiges Endgerät mit einer kontaktlosen Chipkarte kommuniziert.

Passiver Modus als Slave Hierbei generiert das Gerät kein eigenes Feld sondern verhält sich wie eine kontaktlose Chipkarte, weswegen der Modus auch Card-Emulation-Modus genannt wird. Dieser Fall ist für bargeldlose Bezahlvorgänge interessant, in denen zum Beispiel ein NFC-fähiges Smartphone die Kreditkarte ersetzen soll. Der Modus spielt später bei der Vorstellung der Lösung eine wichtige Rolle.

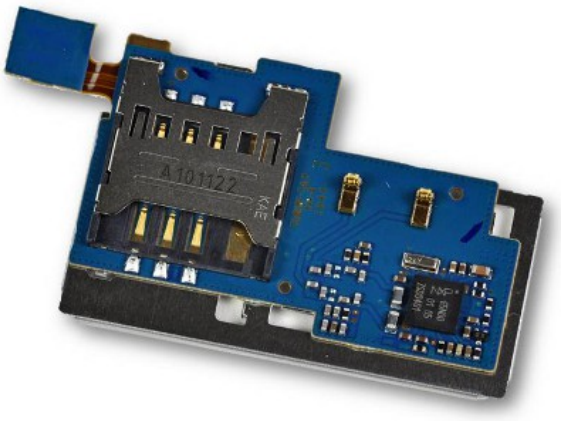

Abbildung 2.2.: Platine des Nexus S mit NXP PN65N - Quelle: [13]

Sicherheitsarchitektur Eine der angestrebten Hauptanwendungen von NFC stellt das bargeldlose Bezahlen mit mobilen Endgeräten dar, was ein hohes Maß an Sicherheit erfordert. Die kurze Reichweite von NFC kommt dem hierbei zu Gute, da dadurch unbeabsichtigte Verbindungen und Man-in-the-Middle Angriffe nahezu ausgeschlossen sind. Trotzdem

können sensible Empfangsgeräte die NFC-Kommunikationen noch in mehreren Metern Entfernung mitlesen, weswegen die Daten optional mit Hilfe von Secure Messaging verschlüsselt und signiert werden können. Ein weiteres Problem stellt die Architektur innerhalb des Endgeräts dar. Diese verfügen heutzutage über mächtige aber dadurch auch angreifbare Betriebssysteme, welche nicht die Anforderungen erfüllen, um sensible Daten zu beherbergen. Hierbei bedient man sich nun wieder der Chipkartentechnologie und baut Sicherheitschips (Secure Elements) so in die Endgeräte ein, dass sie über eine direkte Verbindung zum NFC-Controller verfügen. Ein Beispiel ist der PN65N Chip von NXP, der einen NFC-Controller und ein Secure Element beinhaltet. Es wird zum Beispiel im Nexus S Smartphone verbaut, wie in Abbildung 2.2 zu sehen ist. Über die Platine ist der Chip auch direkt mit der SIM-Karte verbunden.

Vergleich der Übertragungsarten

Alle hier vorgestellten Übertragungsarten haben die drahtlose Übertragung von Daten als Gemeinsamkeit, aber unterscheiden sich hauptsächlich hinsichtlich der Reichweite, Datenrate und Sicherheitsmechanismen. Bezüglich der Anforderungen für den Transport sensibler Informationen kann man zwischen zwei Anwendungsfällen unterscheiden:

(a) Die Kommunikation zwischen Endgerät und Datenquelle erfolgt nicht über das Internet. Dies ist zum Beispiel der Fall, wenn die Übertragung einer einer vertrauenswürdigen Umgebung stattfindet. Eine hohe Reichweite kann hier eher hinderlich sein, da einem Angreifer in der Umgebung das Abhören und Beeinflussen der Verbindung erleichtert wird.

(b) Die Kommunikation findet über das Internet statt. Dies dürfte der häufiger auftretende Anwendungsfall sein, bei dem zwischen den Kommunikationspartnern weitere unbekannte Zwischenstationen existieren, weswegen die Sicherheitsmechanismen der kontaktlosen Schnittstelle hier weniger eine Rolle spielen.

Für den Fall (a) bieten sich, wie bereits genannt, Übertragungsarten mit einer geringen Reichweite an, die eine direkte Kommunikation zwischen Sender und Empfänger unterstützen, damit so wenig Akteure wie möglich involviert sind. Aus diesen Gründen scheidet die Mobilfunkschnittstelle hier aus. Auf der anderen Seite erfüllen NFC und kabelgebundene Techniken, wie USB, diese Anforderungen vollständig. Zusätzlich ist über NFC eine direkte Kommunikation mit dem Secure Element möglich, sodass der Weg über das unsichere Betriebssystem[5] des Endgerätes vermieden werden kann. Bei WLAN und Bluetooth können die Kommunikationspartner ebenfalls direkt miteinander kommunizieren, jedoch müssen auf Grund der hohen Reichweite weitere Sicherheitsmaßnahmen ergriffen werden, um das Abhören oder Verändern der Daten zu verhindern.

[5] Die Sicherheitsmechanismen und Schwachstellen von Android werden in Abschnitt 2.1.2 näher betrachtet.

Bei Fall (b) sind die Sicherheitsanforderungen an die drahtlose Verbindung weniger hoch als bei (a). Die drahtlose Verbindung stellt nur einen kleinen Abschnitt der gesamten Kommunikation dar, weswegen die Sicherheit der Daten(pakete) selbst wichtiger ist. Wichtiger als die Sicherheit ist die schnelle Datenübertragung und die Möglichkeit, das Endgerät über diese Schnittstelle mit dem Internet verbinden zu können. Da das bei Bluetooth[6] und NFC eher unüblich ist und sie außerdem nur eine geringe Übertragungsrate bieten, kommen sie hier nicht in Betracht. Mobilfunk und WLAN eignen sich stattdessen beide gleich gut.

2.1.2. Android als ein mobiles Betriebssystem

Android [1] ist ein auf Linux basierendes Betriebssystem, das hauptsächlich in mobilen Endgeräten Verwendung findet und im Oktober 2008 veröffentlicht wurde. Es wird von der Open Handset Alliance entwickelt, in der Google eine tragende Rolle spielt. Auch wenn für mobile Endgeräte weitere Betriebssysteme, wie iOS oder Symbian, existieren, wird aus folgenden Gründen Android in der vorliegenden Arbeit untersucht:

- Es ist sehr weit im mobilen Bereich verbreitet. Im vierten Quartal 2011 basierten über 50% aller verkauften Smartphones auf Android [31]. Daraus resultiert, dass das Betriebssystem von vielen Anbietern für Smartphonezubehör, unter anderem für Secure Elements im microSD-Format, unterstützt wird.
- Es steht zum Großteil unter der Apache 2.0 Lizenz[7] und ist somit Open Source.
- Es besitzt eine mehrschichtige Sicherheitsarchitektur, in der die Programme klar voneinander isoliert sind. Genau genommen wird jedes Programm in einer eigenen Sandbox ausgeführt, die durch ein eigenes Linux Benutzerkonto realisiert wird. Da sie eine wichtige Rolle bei der Entwicklung von sicherheitskritischen Anwendungen spielt, wird die Sicherheitsarchitektur später im aktuellen Abschnitt näher betrachtet.
- Für die Entwicklung von Androidprogrammen werden kostenlose Werkzeuge zur Verfügung gestellt. Zusätzlich lassen sich zu Testzwecken Programme auch ohne einen kostenpflichtigen Entwickleraccount unter Android ausführen.

Diese Gründe lassen hoffen, dass eventuell auftretende Hürden bei der Anwendungsentwicklung geringer ausfallen, als es bei anderen Betriebssystemen der Fall wäre. Im Folgenden soll Androids Systemaufbau näher betrachtet werden. Hierbei werden dessen Grenzen aufgezeigt, wobei deutlich wird, dass der normale Telefonspeicher nicht sicher genug für die Speicherung von Identitätstoken ist. Deswegen wird mit seek-for-android eine API vorgestellt, mit der unter Android auf Secure Elements und die SIM-Karte zugegriffen werden kann.

Wie in Abbildung 2.3 zu sehen ist, teilt sich das System in folgende drei Schichten auf:

[6] Der Fall, dass ein PC über Bluetooth mit einem Smartphone verbunden wird, um dessen Mobilfunkschnittstellen nutzen zu können, wird hier nicht betrachtet, da eine Kopplung von zwei mobilen Endgeräten für diesen Einsatzzweck untypisch ist.

[7] `http://source.android.com/source/licenses.html`

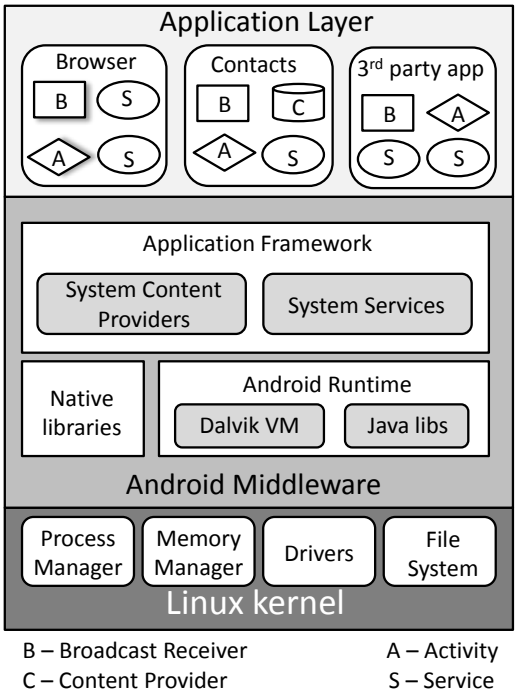

Abbildung 2.3.: Android Architektur – Quelle: [28]

Der Linuxkernel verwaltet die Hardwarekomponenten und stellt sie den darüber liegenden Schichten über Schnittstellen zur Verfügung.

Die Middleware kapselt die Kernelschicht und stellt Bibliotheken und Dienste für Anwendungen zur Verfügung. Zu ihr gehört auch die Dalvik Virtual Machine (Dalvik VM), bei der es sich um eine virtuelle Maschine zur Ausführung von Android-Anwendungen handelt[8].

Die Applikationsebene enthält normale Anwendungen, die mit Nutzerrechten laufen und auf native Bibliotheken sowie auf das Application Framework zugreifen können.

Android-Anwendungen werden zum Großteil in Java[9] programmiert und anschließend in einen Bytecode konvertiert, der von der Dalvik Virtual Machine ausgeführt werden kann. Sie setzen sich aus unterschiedlichen Komponenten zusammen. Eine *Aktivität* stellt ein Fenster dar, mit dem der Benutzer interagiert. Soll eine Anwendung mehrere Fenster besitzen, so besteht sie aus mehreren Aktivitäten, die lose miteinander verbunden sind. Da in Android zu einem Zeitpunkt immer nur eine Aktivität aktiv ist, wird beim Starten einer neuen Aktivität die vorherige pausiert und im „back stack“ abgelegt. Dabei handelt es sich um einen Last-in-First-out Speicher, sodass bei einem Druck auf „zurück“ die zuletzt

[8] `http://code.google.com/p/dalvik`

[9] Mit dem Android NDK können Programmteile auch nativ in z. B. C oder C++ programmiert werden. Auf der Entwicklerseite wird jedoch dazu geraten, nur in sich abgeschlossene CPU-intensive Funktionen so umzusetzen. Siehe dazu `http://developer.android.com/tools/sdk/ndk/index.html`

abgespeicherte Aktivität ausgeführt wird. Prozesse, die durchgängig im Hintergrund laufen sollen und keine Nutzerinteraktion erfordern, werden mit *Services* realisiert. Anwendungen können einen Service an sich binden und per Interprozesskommunikation auf von ihm zur Verfügung gestellte Funktionen zugreifen. *Content Provider* bieten einen strukturierten Zugriff auf Daten für andere Anwendungen. Über einen *Broadcast Receiver* können Nachrichten von anderen Prozessen empfangen werden.

Sicherheitsarchitektur

Androids Sicherheitsarchitektur basiert auf den Linux Zugriffsbeschränkungen, die um weitere Schichten erweitert wurden. Anwendungen laufen isoliert in einer eigenen Sandbox. Außerdem wird für jede Anwendung eine eigene Linux-Benutzer-ID angelegt, über die die Zugriffsrechte für die Anwendungsdateien gesteuert werden. Unter diesem Benutzerkonto läuft für jede Anwendung eine eigene Instanz der Dalvik VM als eigenständiger Prozess, in der die Anwendung ausgeführt wird. Werden Applikationen vom selben Hersteller signiert, können sie sich eine Benutzer-ID teilen und in derselben virtuellen Maschine laufen, weswegen sie gegenseitigen Zugriff auf ihre Daten besitzen. So abgeschottete Anwendungen laufen normalerweise nur mit Benutzerrechten.

Für Interprozesskommunikation müssen Anwendungen während der Installation spezielle Berechtigungen einfordern, von denen in der Android-API einige vorgegeben sind. Applikationen, die Schnittstellen zur Verfügung stellen, können auch zusätzlich eigene Berechtigungen definieren. Stimmt der Benutzer dem zu, können Anwendungen über verschiedene Arten miteinander kommunizieren. Neben dem direkten Aufrufen von Java-Funktionen, können spezielle Datenstrukturen, sogenannte *Intents*, versendet werden. Diese enthalten neben den Daten unter anderem eine abstrakte Beschreibung der aufzurufenden Operation. Intents können direkt an Aktivitäten, Diensten oder an eine Menge von Broadcast Receivern gesendet werden. Eine weitere Möglichkeit bietet die Kommunikation über Sockets, für die ebenfalls zum Installationszeitpunkt eine Berechtigung angefordert werden muss. Außerdem können Anwendungen auch über Dateien miteinander kommunizieren. Speichert eine Anwendung Daten im Dateisystem, kann sie bestimmen, ob sie die Dateien in einem privaten Verzeichnis im Telefonspeicher, auf das nur Anwendungen mit der gleichen Benutzer-ID zugreifen können, oder in einem öffentlichen Verzeichnis speichert. Die Vielzahl an Möglichkeiten der Interprozesskommunikation unter Android erschwert es dem Benutzer allerdings, den Informationsfluss abzuschätzen, wenn er Prozessen bestimmte Berechtigungen einräumt. Da die Sicherheitsarchitektur noch weitere Schwachstellen besitzt, sollen diese im Folgenden betrachtet werden.

Grenzen

Eine Anwendung muss während der Installation alle benötigten Berechtigungen vom Benutzer anfordern, der diese nur erteilen oder die Installation abbrechen kann. Einzelne Rechte

können also nicht verweigert werden[10]. Auch sind diese teilweise sehr weit gefasst, sodass Anwendungen meistens mehr Rechte erhalten, als sie eigentlich benötigen. Problematisch werden solche zu weit gefassten Berechtigungen durch privilege Escalation Angriffe. Dabei greift ein Prozess ohne bestimmte Rechte einen Prozess an, der diese Rechte besitzt. Ist der privilegierte Prozess anfällig gegenüber bestimmten Angriffen, kann der angreifende Prozess somit eventuell Rechte erlangen, die er vorher nicht besaß.

Auch wenn Benutzer normalerweise keine root-Rechte besitzen, existieren Möglichkeiten, sich diese auf einem Android-Gerät zu beschaffen. Das führt jedoch zu zwei Problemen:

- Wenn ein Benutzer sich root-Rechte auf einem Gerät beschaffen kann, kann ein Angreifer dies eventuell ebenfalls. Schadprogramme mit Rootkittechnik können sich vor Virenscannern und dem Benutzer verbergen und somit unbemerkt Daten sammeln, löschen oder modifizieren.
- Werden root-Rechte an Anwendungen vergeben, werden alle Sicherheitsmechanismen von Android ausgehebelt. Schadprogramme können somit auf einem Gerät alle Funktionen ausführen, ohne weitere Rechte einfordern zu müssen.

Da es sich bei Android um eine Rich Execution Environment (REE), also ein komplexes Betriebssystem, handelt, sind Fehler unvermeidbar. Erscheint eine neue Version, dauert es jedoch im Schnitt neun Monate, bis sie von den Herstellern für die einzelnen Geräte zur Verfügung gestellt wird [41]. Problematisch ist ebenfalls, dass manche Geräte zu wenig Kapazitäten besitzen und keine Updates mehr erhalten.

Alles in allem ist es zu unsicher, Identitätstoken aus einer sicheren Datenquelle ungeschützt im Speicher eines Androidgerätes abzulegen, nicht zuletzt da die Anzahl an Schadprogrammen mit Rootkittechnik und gültiger Signatur stetig zunimmt [27]. Eine Möglichkeit wäre es, die Token extra zu verschlüsseln, wobei der Schlüssel ebenfalls nicht ungeschützt im Gerät gespeichert werden darf. Hier bietet es sich an, den Schlüssel entweder mit Hilfe eines Passwortes ebenfalls zu verschlüsseln oder ihn in einem sicheren Schlüsselspeicher wie dem SE oder der UICC zu speichern, für die jedoch eine extra API benötigt wird.

Seek for Android

Ein SmartCard API, das gute Chancen hat, in den Android Kernel aufgenommen zu werden, ist Seek for Android [4]. Es implementiert die von der SIMalliance spezifizierte Open Mobile API [5] und stellt Funktionen bereit, um auf SE oder UICC zugreifen zu können. Android Applikationen können somit über eine einheitliche Schnittstelle per APDUs mit den Chipkarten kommunizieren, wodurch die Entwicklung wesentlich vereinfacht wird. Die Sicherheitsmechanismen basieren auf Androids Sicherheitsarchitektur. Programme, die die API benutzen wollen, müssen während der Installation eine extra Berechtigung vom

[10] In einer aktuellen Arbeit wird eine Lösung präsentiert, mit der Anwendungen einzelne Zugriffsrechte entzogen werden können. Dafür wird in ihren Programmcode eingegriffen und die kritischen Aufrufe gekapselt[25].

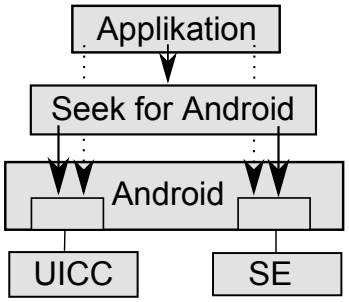

Abbildung 2.4.: Seek for Android Übersicht

Benutzer einfordern. Leider ist Seek for Android ebenfalls für die eben genannten genannten Angriffe anfällig. Besitzt ein Schadprogramm root-Rechte, so kann es die Chipkartenkommunikation angreifen und z. B. PIN-Eingaben zur Freischaltung eines Secure Elements abfangen. Eine sichere Ein- und Ausgabe ließe sich theoretisch durch eine sog. *Trusted Execution Environment* (TEE) realisieren. Dabei handelt es sich um einen abgeschotteten Bereich im Prozessor, der vertrauenswürdig mit der Smartphoneperipherie, also Tastatur und Display, kommuniziert. Obwohl diese Technologie bereits durch GlobalPlatform spezifiziert [19] und mit der TrustZone von ARM[11] auf Hardwareebene umgesetzt wurde, befindet sie sich auf der Softwareebene noch in der Entwicklung und wird hier nicht weiter betrachtet.

2.2. Secure Element

Das in Abschnitt 2.1.1 betrachtete NFC-Protokoll ist auch für sicherheitskritische Anwendungen wie das kontaktlose Bezahlen gedacht, für das ein *Secure Element* (SE) benötigt wird. Als Secure Element werden spezielle Chips im Umfeld von mobilen Endgeräten bezeichnet, die Ähnlichkeiten mit Chipkarten, insbesondere Prozessorkarten, aufweisen. Tatsächlich unterscheiden sich SE und Prozessorkarten meistens nur im Formfaktor basieren aber auf ähnlichen Chips und können somit programmiertechnisch gleich behandelt werden. Auf Grund der vielen Ähnlichkeiten werden die Begriffe Secure Element, Chipkarte und Prozessorkarte synonym verwendet, sofern die Hardwareumsetzung nicht von Belang ist. Secure Elements sind in der Regel sicherheitszertifiziert. So ist z. B. beim „Mobile Security Card SE 1.0“ von Giesecke & Devrient der Chip nach Common Criteria (CC) EAL 5+ zertifiziert, was bedeutet, dass er semiformal entworfen und getestet wurde. Allerdings muss beachtet werden, dass das Betriebssystem eines SE meistens ein geringeres Sicherheitslevel als die Hardware des Chips aufweist. Sm@rtCafé Expert 5.0, das Betriebssystem des genannten SE, ist ebenso wie das JavaCard Betriebssystem JCOP von NXP nur nach CC EAL 4+ zertifiziert.

[11] `http://www.arm.com/products/processors/technologies/trustzone.php`

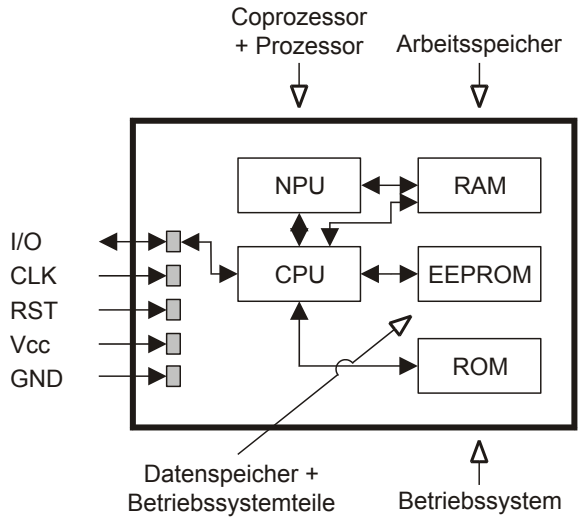

Abbildung 2.5.: Typischer Aufbau einer Prozessorkarte - Quelle: [42]

Secure Element und Prozessorkarte besitzen den gleichen schematischen Aufbau, der in Abbildung 2.5 zu sehen ist. Eine wesentliche Komponente stellt die CPU dar, bei der es sich trotz diverser Fortschritte in den letzten Jahren oft immer noch um einen 8-Bit Mikroprozessor handelt (Quelle: [42] S. 83 ff.). Da für die sicherheitskritischen Anwendungen oft komplexe kryptografische Berechnungen durchgeführt werden müssen, besitzen SE in der Regel einen Coprozessor. In dieser NPU (Numerical Processor Unit) sind die wichtigsten Signatur- und Verschlüsselungsalgorithmen sowie Hashfunktionen und Zufallszahlengeneratoren in Hardware implementiert und können somit um Größenordnungen schneller durchgeführt werden, als auf der CPU. Wichtige Bestandteile des Betriebssystems werden während der Herstellung im ROM gespeichert und sind dadurch vor nachträglichen Änderungen geschützt. Allerdings muss sich bereits während der Produktion auf ein Betriebssystem und eine Laufzeitumgebung für die Programme festgelegt werden. Restliche Teile des Betriebssystems befinden sich zusammen mit den Programmen und dauerhaft zu speichernden Daten im EEPROM. Dabei handelt es sich um einen elektrisch programmierbaren Speicher, der Daten auch ohne Stromzufuhr über mehrere Jahre halten kann. Der Speicher bietet mit gängigen Größen um die 75 KB vergleichsweise viel Platz. Auf ihn kann schreibend jedoch nur sehr langsam zugegriffen werden. Einen ungefähr um den Faktor 1000 schnelleren Schreibzugriff bietet der RAM, von dem jedoch meistens nicht einmal 2 KB zur Verfügung stehen. Auch muss beachtet werden, dass der RAM-Inhalt bei Wegfall der Stromzufuhr verloren geht.

Aufgrund der Ähnlichkeiten kann ein Großteil der Spezifikationen und Standards für Chipkarten auch für Secure Elements verwendet werden. Die wichtigsten sind dabei ISO/IEC 7816, GlobalPlatform und JavaCard. In der ISO/IEC 7816 wird unter anderem die Schnittstelle nach außen spezifiziert. Dabei handelt es sich um eine kontaktbehaftete serielle Halbduplexverbindung mit einer in der Regel maximalen Datenrate von 223 kbit/s. Manche SE

können auch über eine kontaktlose Verbindung nach ISO/IEC 14443 angesprochen werden, bei der eine Datenrate in derselben Größenordnung erreicht wird[12]. JavaCard und GlobalPlatform werden in den Abschnitten 2.2.2 und 2.2.3 näher betrachtet.

2.2.1. Secure Element Integration in mobilen Endgeräten

Secure Elements können, wie in Abbildung 2.6 zu sehen ist, auf verschiedene Weise in mobile Endgeräte integriert werden. Am bekanntesten ist die SIM-Karte. Diente sie früher nur der Identifizierung von Mobilfunkteilnehmern, so kann sie mittlerweile auch Anwendungen von Drittanbietern ausführen und somit als SE angesehen werden. Weiterhin haben sich Secure Elements im microSD-Format und in das Endgerät integrierte Sicherheitscontroller etabliert. Der in der Abbildung ebenfalls erkennbare NFC Controller wird häufig zusammen mit einem Secure Element verwendet. Ein Grund dafür ist, dass Anwendungen, die ein Secure Element benötigen, häufig auch NFC verlangen und umgekehrt.

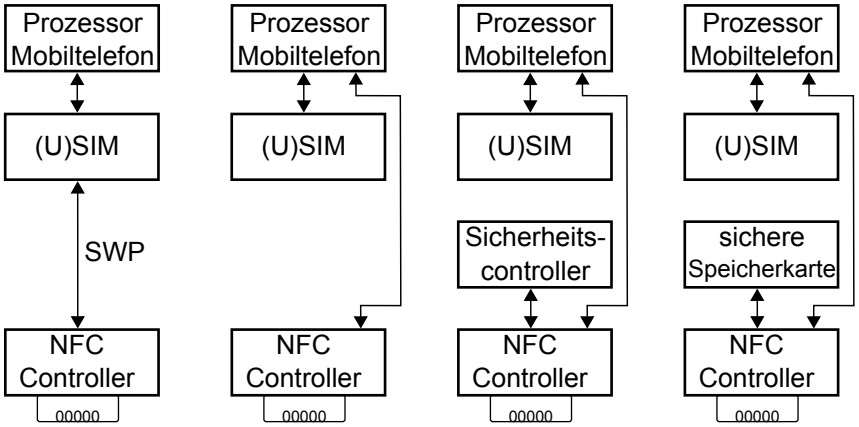

Abbildung 2.6.: Arten der Secure Element Integration - Quelle: [42]

SIM-Karte

Bei der Subscriber Identity Module (SIM)-Karte handelt es sich um eine austauschbare SmartCard, die jedes GSM-fähige mobile Endgerät benötigt, um sich in das Mobilfunknetz einzuwählen. In der Spezifikation EN 300 920 wird sie wie folgt definiert: „Die SIM ist eine Entität, welche die Identität des Teilnehmers enthält. Die primäre Funktion der SIM ist die Sicherstellung der Echtheit der Mobilstation gegenüber dem Netzwerk“ [13]. Dabei authentisiert sie das Endgerät nach einer erfolgreichen PIN-Eingabe gegenüber dem ihr

[12] Die Verbindung kann dabei entweder indirekt über einen NFC-Chip, wie z. B. beim bereits genannten PN65N Chip von NXP oder direkt wie zum Beispiel bei der SE microSD Karte In2Pay von Device Fidelity erfolgen (Siehe dazu `http://www.devifi.com/assets/IN2PAY_MICROSD_V2.1.pdf`).

[13] Übersetzung aus [42]

zugehörigen Mobilfunkanbieter. Für die Authentisierung werden eine International Mobil Subscriber Identity (IMSI) sowie Schlüsselmaterial benötigt, die sich zusammen mit weiterem Schlüsselmaterial zur Verschlüsselung von Gesprächen im sicheren Speicher der SIM-Karte befinden. Die SIM-Karte bietet noch weitere Funktionen, wie das SIM Application Toolkit (SAT), über das hardwaregeschützte Mehrwertdienste auf der SIM-Karte angeboten werden können. Zum Aufspielen von eigenen Applikationen auf die SIM ist jedoch die Zusammenarbeit mit dessen Eigentümer, also dem entsprechenden Mobilfunkanbieter, erforderlich. Bei dem Mobilfunkstandard der dritten Generation (3G), in Deutschland unter UMTS bekannt, wurde das Multiapplikationskonzept auf der SIM-Karte noch weitergeführt. Die SIM wurde in Universal Subscriber Identity Module (USIM) umbenannt und stellt nun nicht mehr die komplette Chipkarte, sondern vielmehr nur noch eine Applikation auf dieser dar[14]. Außerdem wurde die Karte in Universal Integrated Circuit Card (UICC) umbenannt.

Aufgrund der vielen Funktionen und Akteure existieren eine Reihe von Standards, Normen und Spezifikationen, unter anderem von dem Europäischen Institut für Telekommunikationsnormen (ETSI), dem 3rd Generation Partnership Project (3GPP), der International Organization for Standardization (ISO) und GlobalPlatform. In der 3GPP TS 03.48 wird z. B. das Over-The-Air (OTA) Provisioning von Anwendungen in die SIM-Karte spezifiziert. Dabei wird eine sichere Ende-zu-Ende Verbindung zwischen MNO und SIM-Karte aufgebaut, worüber eine Anwendung übertragen und anschließend im UICC installiert wird. Zur Übertragung können spezielle SMS oder das Bearer Independent Protocol (BIP) verwendet werden.

In ETSI TS 102 613 und TS 102 622 wird mit dem Single Wire Protocol (SWP) ein kontaktloses Interface der SIM-Karte beschrieben, worüber sie direkt mit dem NFC-Chip des Geräts verbunden ist. Somit kann sichergestellt werden, dass mögliche Schadsoftware auf dem Betriebssystem des Endgeräts nicht die NFC-Kommunikation mithören kann.

Alles in allem eignet sich die SIM-Karte von der technischen Seite her sehr gut zur sicheren Speicherung von Identitätstoken. So kann sie eindeutig einer Person zugeordnet werden und Funktionen, wie das USAT zum Hosten und OTA-Mechanismen zum Aufspielen von Applikationen, verringern den Aufwand. Jedoch benötigt ein Applethersteller die Unterstützung des Mobilfunkanbieters um ein Applet in dessen SIM-Karte installieren zu können. Da es sehr viele solcher Anbieter gibt, für die jedoch keine zentrale Anlaufstelle existiert, ist die Nutzung der SIM-Karte somit mit einem erheblichen Mehraufwand verbunden.

Integriertes Secure Element

Bei den Smartphoneherstellern gibt es in letzter Zeit Bestrebungen, die Funktionen der SIM-Karte direkt in das Telefon zu integrieren, um damit auf eine zusätzliche Karte verzichten zu können[15]. Auch wenn sich die Idee bis jetzt noch nicht durchsetzen konnte,

[14] Das SIM Application Toolkit wurde folglich ebenfalls in USIM Application Toolkit (USAT) umbenannt.
[15] `http://www.golem.de/1011/79495.html`

werden bereits Secure Elements in manchen Smartphones verbaut[16]. Diese gehen meist mit NFC einher, um das kontaktlose Bezahlen mit einem Smartphone zu ermöglichen. Ende 2011 veröffentlichte Google mit Google Wallet[17] solch ein Bezahlsystem, das ein Secure Element zur Verwaltung des Schlüsselmaterials benötigt. Tatsächlich bietet ein fest eingebautes SE mit NFC-Interface auch gute technische Grundlagen zur sicheren Speicherung einer elektronischen ID. Allerdings ist es aus organisatorischer Sicht schwer, an das Schlüsselmaterial für das SE zu gelangen, welches in der Regel unter der Kontrolle des Herausgebers liegt.

Secure Element im microSD-Format

Einige Hersteller, wie Giesecke & Devrient[18] oder Device Fidelity[19], bieten sichere Speicherkarten im microSD-Format an. Diese beinhalten neben normalen Flash-Speicher auch ein Secure Element, das in der Regel sicherheitsevaluiert und kompatibel zu der GlobalPlatform Kartenspezifikation und JavaCard ist. Durch das microSD-Format lassen sich so SE bei fast allen neuen Smartphones nachrüsten. Da die Karten mit Standardschlüsseln ausgeliefert werden, können beliebige Applets auf ihnen installiert werden, sodass sie sich gut zum Testen eignen. Wird die Karte an Dritte weitergegeben, können die Standardschlüssel durch eigene ersetzt werden. Nachteilig ist, dass die Advanced Security SD Card (ASSD) Spezifikation[20] zur Kommunikation mit dem SE über das microSD Interface noch nicht auf allen Betriebssystemen unterstützt wird, sodass man bei der Programmierung eventuell auf das SDK des Herstellers angewiesen sein kann. Auf Seiten des Endgerätes wird die Spezifikation zum Beispiel durch Seek for Android implementiert. In Abschnitt 4.3 wird noch gesondert auf In2Pay microSD von Device Fidelity eingegangen.

2.2.2. Java Card

Bei Java Card[21] handelt es sich um eine Ausführungsumgebung für Programme auf Chipkarten. Es stellt eine Teilmenge von Java dar, um den geringen Ressourcen von Chipkarten gerecht zu werden. So gibt es z. B. keine automatische Speicherbereinigung und die API ist stark eingeschränkt. Die Java Card Virtual Machine (JCVM) ist sehr stark mit der Hardware verbunden. Sie wird bereits während der Kartenherstellung neben dem Betriebssystem fest in die Karte eingebrannt und startet automatisch, sobald die Karte mit Strom versorgt wird. Erst wenn der Lebenszyklus der Karte sein Ende erreicht hat, die Karte also gesperrt ist, endet auch der Lebenszyklus der JCVM.

[16] http://www.nxp.com/news/press-releases/2011/11/nxp-nfc-solution-implemented-in-galaxy-nexus-from-google.html
[17] http://www.google.com/wallet
[18] http://www.gd-sfs.com/de/die-mobile-security-card/mobile-security-card-se-1-0
[19] http://www.devifi.com/in2pay_microsd.html
[20] https://www.sdcard.org/developers/overview/ASSD
[21] http://www.oracle.com/technetwork/java/javacard/overview/index.html

Durch Java Card ist es möglich, ein Applet nachträglich auf eine Chipkarte zu laden, die sich bereits im Feld befindet. Dafür wird das Java-Card-Applet nach dem normalen Kompiliervorgang zusätzlich komprimiert und über APDUs auf die Karte gespielt. Nachdem es zuerst instanziiert und danach selektiert wurde, kann es über APDUs nach außen hin kommunizieren.

Sicherheit

Auf der Karte sind die Applets untereinander durch die Applet Firewall der JCVM abgeschirmt, die fast jeden übergreifenden Zugriff untersagt. Applets werden dafür in Kontexte eingeteilt, die sich an den von Java bekannten Packages orientieren. Nur Applets innerhalb des selben Kontextes können Objekte miteinander teilen [29]. Durch den stark eingeschränkten Funktionsumfang von Java Card und dessen Umsetzung durch in der Regel sicherheitsevaluierte Betriebssysteme, kann davon ausgegangen werden, dass ein Java Card Applet in der Regel besser vor Manipulationen geschützt ist, als ein normales Applet in einer REE. Trotzdem muss beachtet werden, dass meist nur sehr wenige öffentliche Informationen über SmartCard Betriebssystem existieren und diese bei Bekanntwerden einer Sicherheitslücke nur schlecht aktualisiert werden können.

JCOP

Bei Java Card OpenPlatform (JCOP) handelt es sich um ein SmartCard Betriebssystem, auch Card Operating System (COS) genannt, das von IBM entwickelt wurde. Der Name setzt sich aus den unterstützten Standards Java Card und GlobalPlatform (früher bekannt als Open Platform) zusammen. Je nach den unterstützten Kryptofunktionen und Interfaces existierten unterschiedliche JCOP-Arten. Häufig verwendet wird zum Beispiel JCOP 41 v2.3.1, das konform mit dem Java-Card-Standard 2.2.1 und GlobalPlatform 2.1.1 ist und unter anderem RSA bis 2432 Bit und Elliptische Kurven Kryptografie unterstützt[22].

2.2.3. GlobalPlatform

GlobalPlatform (GP) [3], das 1999 als Nachfolger von Visa Inc.'s Open Platform gegründet wurde, ist ein internationales Industrieforum, das sich als Ziel gesetzt hat, möglichst weltweit anerkannte Standards im SmartCard Umfeld zu schaffen. Dabei wird das Hauptaugenmerk auf die Verwaltung von Applikationen unterschiedlicher Hersteller auf derselben SmartCard und die Verwaltung von verschiedenen SmartCards im selben System gelegt. GlobalPlatform besteht zur Zeit aus über 75 Mitgliedern aus unterschiedlichen wirtschaftlichen Bereichen. Die entwickelten Spezifikationen werden einem der drei Bereiche *Card*, *Device* oder *Systems* zugeordnet, je nachdem ob es um die Verwaltung innerhalb einer Karte (bzw. SE), eines Gerätes oder des kompletten Systems mit verschiedenen Parteien

[22] `http://www.nxp.com/documents/line_card/75016728.pdf`

geht. Für die vorliegende Arbeit sind vor allem die Karten- und Systemspezifikationen von Interesse, da mit ihrer Hilfe der Lebenszyklus von Secure Element und Applet beeinflusst werden kann.

Karten-Spezifikation

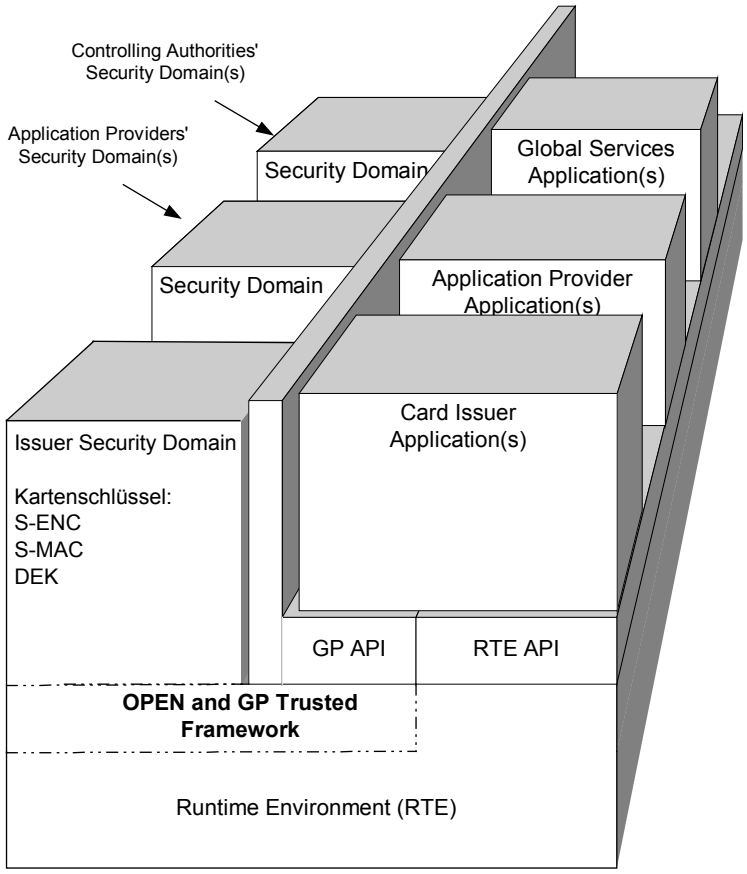

Abbildung 2.7.: GlobalPlatform Kartenarchitektur - nach [18]

In der Karten-Spezifikation wird die Verwaltung mehrerer Applikationen von verschiedenen Anbietern auf einer Karte beschrieben. Aktuell befindet sich die „GlobalPlatform Card Specification“ (GPC-Spec) [18] in der Version 2.2.1, wobei jedoch nur Version 2.1.1 von gängigen SmartCards und Secure Elements unterstützt wird. Die vorliegende Arbeit bezieht sich trotzdem auf die aktuelle Version der Spezifikation, da diese eine bessere Struktur besitzt und somit verständlicher als die ältere Version ist. Auf Änderungen zwischen den beiden Versionen wird an entsprechender Stelle hingewiesen.

In Abbildung 2.7 ist die Aufteilung des Secure Elements nach GlobalPlatform zu sehen. Dabei werden die Applikationen und das Betriebssystem selber nicht verändert, sondern

eine extra GlobalPlatform API (GP API), parallel zur API der Ausführungsumgebung (RTE API) eingeführt, sodass bestehende Kartensysteme leicht erweitert werden können. Die wichtigste Komponente bildet die Security Domain (SD), die als Repräsentation der beteiligten Akteure auf der Karte verstanden werden und verschiedene sicherheitskritische Funktionen ausführen kann. Für die Verwaltung der Security Domains ist das GlobalPlatform Environment (OPEN)[23] zuständig. Es enthält unter anderem die Card Registry, in der alle GP bezogenen Daten, wie die Berechtigungen der SD und die Zuordnungen zwischen SD und Applikationen, gespeichert sind. Soll über eine Security Domain eine Funktion ausgeführt werden, wie zum Beispiel die Sperrung der Karte, entscheidet das OPEN an Hand der Card Registry, ob die Funktion zulässig ist und führt sie dann gegebenenfalls aus. Weiterhin stellt sie über das Cardholder Verification Management eine PIN-Abfrage zur Verfügung, die von einer Applikation zur Verifikation des Benutzers genutzt werden kann.

Lebenszyklus Eine Karte besitzt einen Lebenszyklus, der direkten Einfluss auf ihr Verhalten hat. Dieser stellt die unterschiedlichen Stationen dar, die eine Karte im Laufe ihres Einsatzes durchläuft. Wie in Abbildung 2.8 zu sehen ist, kann der Lebenszyklus durch eine

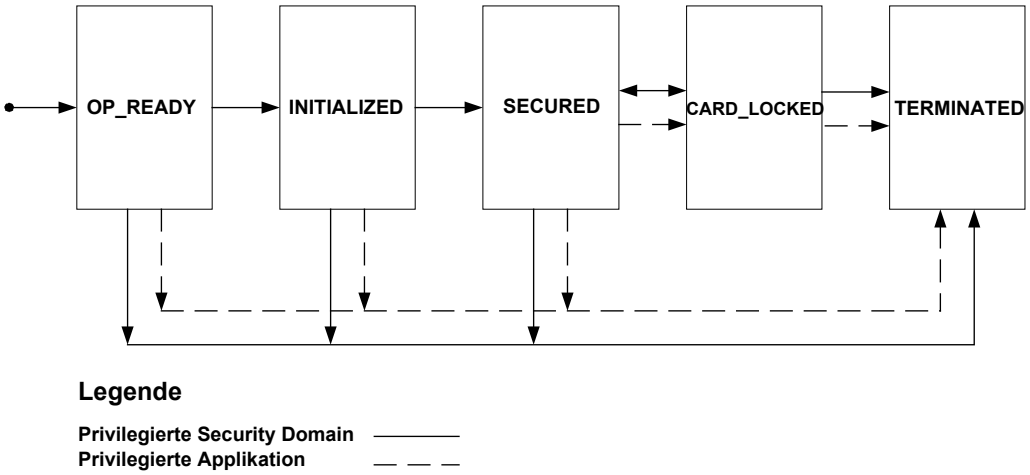

Abbildung 2.8.: Lebenszyklus einer Karte - nach [18]

privilegierte Security Domain oder Applikation beeinflusst werden. Die ersten beiden Status *OP_READY* und *INITIALIZED* werden für die Phase vor der Herausgabe der Karte (Pre-Issuance) verwendet. Dabei ist die Karte bereits soweit einsatzfähig, dass sie personalisiert werden kann und es können Security Domains oder Applikationen de-/installiert werden. Verschiedene Sicherheitsmaßnahmen, wie z. B. Secure-Messaging-Zwang bei der Kommunikation mit einer Security Domain, sind noch nicht aktiv. Nachdem die Kartenschlüssel festgelegt wurden und die Karte ausgeliefert werden soll, wird der Status auf *SECURED* geändert. Die Karte ist nun voll einsatzfähig. Im Status *CARD_LOCKED*

[23] Die Abkürzung kommt durch den ursprünglichen Namen „Open Platform ENvironment“ zustande.

können nur noch Applikationen und Security Domains mit dem Final Application Privileg aufgerufen werden. *TERMINATED* stellt eine Möglichkeit dar, eine Karte permanent zu deaktivieren, wenn z. B. ihre Gültigkeit abgelaufen ist.

Security Domain

Security Domains nehmen als Repräsentationen der jeweiligen Akteure im SE eine zentrale Rolle bei GlobalPlatform ein. Prinzipiell existieren drei Arten von Akteuren, die eine SD besitzen können:

Issuer Der Issuer ist in der Regel der SE-Herausgeber und besitzt alle Berechtigungen auf dem SE. Die Issuer Security Domain (ISD) kann den Lebenszyklus der Karte, der Security Domains oder der Applikationen beeinflussen und neue Applikationen oder Security Domains installieren, sowie alte löschen. In der ISD können unter anderem zwei Daten gespeichert werden, die später in der vorgestellten Lösung verwendet werden.

Die **Issuer Identification Number (IIN)** enthält in der Regel den ISO 7812 konformen Identifizierer des Issuers, um die Karte einem spezifischen Kartenverwaltungssystem zuordnen zu können.

Die **Card Image Number (CIN)** soll eine vom Issuer eindeutig festgelegte Nummer enthalten, um die Karte innerhalb eines Kartenverwaltungssystem eindeutig identifizieren zu können.

Application Provider Der Application Provider entwickelt und betreibt eigene Applikationen. Er benötigt nur dann eine eigene SD, wenn er Schlüsselmaterial verwenden möchte, auf das der Issuer keinen Zugriff haben darf.

Controlling Authority Die Controlling Authority, auch Trusted Service Manager (TSM) genannt, kümmert sich im Auftrag des Issuers um die Kartenverwaltung. Durch Erteilung einer Supplementary Security Domain (SSD) für den TSM kann der Issuer dessen Berechtigungen festlegen und behält trotzdem noch die volle Kontrolle über das SE. Solch eine SSD sollte zumindest über eine der folgenden Privilegien verfügen, um sinnvoll agieren zu können:

Delegated Management Besitzt eine SSD dieses Privileg, können Applikationen hochgeladen, installiert, selektiert, gelöscht und anderen Security Domains zugeordnet werden. Außerdem kann die Card Registry verändert werden. Da es sich bei Delegated Management nur um ein optionales Feature handelt bei der der ISD-Inhaber fast die komplette Kontrolle abgibt, wird es von manchen Kartenbetriebssystemen wie JCOP nicht unterstützt.

DAP Verification Der TSM ist für die Zusicherung der Korrektheit von hochgeladenen Applikationen zuständig. Dafür lässt sich ein Application Provider seine

Applikation vom TSM signieren. Diese Signatur wird auch auch Data Authentication Pattern (DAP) genannt. Lädt der Issuer nun Applikation und DAP auf die Karte, überprüft die SSD des TSM die Signatur mit Hilfe des gespeicherten öffentlichen Schlüssels. Ist die Signatur korrekt, kann die Applikation installiert werden. Die Methode hat gegenüber dem Delegated Management den Vorteil, dass der Issuer die alleinige Kontrolle über das SE behält und es trotzdem für ihn und dem Application Provider nur einen zentralen Ansprechpartner, den TSM, gibt.

Der Zugriff auf eine SD kann entweder programmatisch in einer Applikation oder von Außen über APDUs erfolgen, je nachdem, welche Funktionen benutzt werden.

Eine Applikation ist immer mit genau einer SD assoziiert. Diese übernimmt die Schlüsselverwaltung und stellt verschiedene Kryptofunktionen zur Ver- und Entschlüsselung sowie zur Signaturgeneration und -verifikation zur Verfügung. Des Weiteren kann die Applikation über die SD einen geschützten Datenkanal per Secure Messaging aufbauen. Applikationen eines Herstellers werden in der Regel derselben SD zugeordnet, sodass sie auch Zugriff auf dasselbe Schlüsselmaterial erhalten.

Da es sich bei Security Domains im Prinzip um besonders privilegierte Programme handelt, können sie wie Applikationen selektiert werden und Kommandos entgegennehmen. Je nach Berechtigungen kann es sich um sensible Kommandos handeln, deren Übertragung gesichert erfolgen muss. Dafür besitzt eine SD mindestens drei Schlüssel:

S-ENC wird zum Aufbau einer verschlüsselten Secure Messaging Verbindung benötigt.

S-MAC wird zur Absicherung der Integrität der Verbindung mit Hilfe eines Message Authentication Code (MAC) benötigt.

DEK Mit dem Data Encryption Key werden sensible Daten, wie zum Beispiel symmetrische Schlüssel extra verschlüsselt.

DAP-Verification Key Eine SD mit DAP Verification Privileg benötigt den öffentlichen Signaturschlüssel seines Inhabers zur Verifikation des DAP.

Im Gegensatz zur neuen GPC-Spec, die auch asymmetrische Kryptografie unterstützt, können Security Domains in JCOP zur Zeit nur Triple-DES (3DES) Schlüssel[24] verwenden. Wird eine SD neu installiert, so verwendet sie die Schlüssel der gerade selektierten SD. Um einen eigenen Schlüssel zu erstellen oder einen alten zu verändern, muss eine mit Secure Messaging geschützte Verbindung zu der SD aufgebaut werden, über die der Schlüssel, zusätzlich mit dem DEK verschlüsselt, zur SD übertragen wird. Ist das SE noch nicht initialisiert, enthält also nur Standardschlüssel, muss die Verbindung auf eine andere Weise abgesichert werden. In [17] wird zusätzlich mit dem Pull Modell ein Verfahren eingeführt,

[24] Der Data Encryption Standard (DES) ist ein symmetrischer Verschlüsselungsalgorithmus, der mittlerweile auf Grund der zu kurzen Schlüssellänge als unsicher angesehen wird. Durch dreifaches Anwenden des DES (3DES) mit unterschiedlichen Schlüsseln kann die Länge des Schlüssels jedoch vergrößert werden.

bei dem die Schlüssel im SE erstellt und mit dem öffentlichen Schlüssel des SD-Inhabers verschlüsselt an ihn geschickt werden.

Installation von Applikationen

Bevor ein Anwender ein SE nutzen kann, muss es einen Personalisierungsprozess durchlaufen, in dessen Verlauf unter anderem die Schlüssel der ISD ausgetauscht und das SE in einen sicheren Zustand versetzt werden, sodass zuerst nur noch der Herausgeber Zugriff besitzt. Optional kann der Herausgeber bereits vor der Auslieferung eine SSD für den zuständigen TSM anlegen. Soll ein Applet auf einem so gesicherten SE installiert werden, müssen verschiedene Schritte durchlaufen werden, die in [20] beschrieben sind.

Es wird angenommen, dass bereits eine Verbindung zu dem zuständigen Issuer bzw. TSM besteht. Dieser kennt die Eigenschaften des SE und überprüft, ob es zu der aufzuspielenden Applikation kompatibel ist. Sind die Voraussetzungen gegeben, wird wenn nötig eine Security Domain für den Application Provider angelegt und personalisiert und anschließend die Applikation hochgeladen und installiert.

2.3. Authentifizierung im Internet

Während im Englischen nur das Wort „authentication“ existiert, wird im Deutschen zwischen Authentifizierung und Authentisierung unterschieden. Das BSI Glossar meint dazu: „Mit Authentisierung wird dann die Vorlage eines Nachweises zur Identifikation bezeichnet, mit Authentifizierung die Überprüfung dieses Nachweises“ [10]. Mit der Besonderheit wird in der vorliegenden Arbeit allerdings wie im Glossar umgegangen: „Um den Text verständlich zu halten, verzichtet der IT-Grundschutz auf diese Unterscheidung“. Der Begriff der **Authentisierung** wird im Glossar wie folgt definiert: „Authentisierung bezeichnet den Nachweis eines Kommunikationspartners, dass er tatsächlich derjenige ist, der er vorgibt zu sein. Dies kann unter anderem durch Passwort-Eingabe, Chipkarte oder Biometrie erfolgen“. Die genannten Authentisierungsverfahren können auch zusammen verwendet werden. Bei der 2-Faktor-Authentifizierung wird häufig Besitz und Wissen kombiniert, wie zum Beispiel bei der eID-Funktion des neuen Personalausweises. Der Benutzer *besitzt* den nPA, für dessen Benutzung er eine sechsstellige PIN *wissen* muss. Weitere Authentisierungsmöglichkeiten sind das Können von etwas (z. B. Unterschrift), der Ort (z. B. in einem Hochsicherheitsbereich) und die bereits erwähnte Biometrie.

Weitere für diese Arbeit relevante Begriffe werden im Folgenden erklärt:

(Digitale) Identität Die digitale Identität ist die Menge an Daten, durch die eine Person in einem bestimmten Zusammenhang eindeutig bezeichnet und von anderen unterschieden werden kann.

Identitätstoken Ein Identitätstoken oder auch Softtoken genannt, besteht aus signierten Daten, die zur Authentifizierung einer Entität verwendet werden. Zusätzlich kann ein Identitätstoken noch weitere Attribute beinhalten, die bei der Authentifizierung mit übertragen werden. Ein U-Prove Token ist ein spezieller Softtoken, der noch über weitere, datenschutzfreundliche Eigenschaften verfügt. Damit ein U-Prove Token verwendet werden kann, sind zusätzliche private Komponenten, wie zum Beispiel ein privater Schlüssel, notwendig.

Identifikation Die Identifikation dient der eindeutigen Bestimmung einer Person und steht im Gegensatz zur Anonymität.

Anonymität/Pseudonymität Ist eine Person anonym, kann sie nicht identifiziert werden. Ein Pseudonym dient der Verschleierung einer Indentität. Nutzt eine Person ein Pseudonym mehrmals, kann sie jedoch wiedererkannt werden.

eID-System Dabei handelt es sich um ein System, mit dem ein oder mehrere Identitäten verwaltet und zur Authentifizierung im Internet verwendet werden können. Neuere eID-Systeme haben datenschutzfreundliche Merkmale, wie Pseudonymität oder Anonymität, die die eindeutige Identitifikation eines Benutzers in bestimmten Situationen verhindern.

2.3.1. Beispiel: eID-Funktion des nPA

Ein Beispiel für ein eID-System der neuesten Generation ist das deutsche eID-System, das am 1. November 2010 zusammen mit der Einführung des neuen Personalausweises in Betrieb ging. Sofern die eID-Funktion des Ausweises aktiviert ist, kann sich der Inhaber damit im Internet authentisieren. Für die Arbeit ist diese Funktion aus mehreren Gründen relevant. Es ist denkbar, dass die beim nPA verwendeten Sicherheitsmechanismen auch auf eine Lösung mit einem SE übertragbar sind. Des Weiteren eignet sich der nPA als eine authentische Datenquelle aus der die Identitätstoken abgeleitet werden können. Zu guter Letzt wird in dem Abschnitt begründet, wieso ein zum nPA alternatives System in der Arbeit benötigt wird.

Die einzelnen Komponenten des deutschen eID-Systems müssen bestimmten Anforderungen genügen, die in verschiedenen technischen Richtlinien des BSI festgelegt werden. So wird auf Anwenderseite neben einem zur TR-03112 [23] kompatiblen Client (z.B. die AusweisApp) auch ein auf der ISO/IEC 14443 [7] basierender und nach TR-03119 [21] zertifizierter Kartenleser benötigt. Des Weiteren wird für das Auslesen der Ausweisdaten ein nach TR-03130 [15] zertifizierter eID-Server benötigt, wobei die Kommunikation mit dem Ausweis in der TR-03110 [14] spezifiziert wird. Möchte ein Diensteanbieter einen Benutzer authentifizieren und verfügt über ein entsprechendes Berechtigungszertifikat, kann er dem Nutzer eine spezielle Anfrage schicken, die den im Folgenden beschriebenen Prozess anstößt:

(a) Der Benutzer möchte einen Dienst nutzen, für den er sich authentisieren muss.

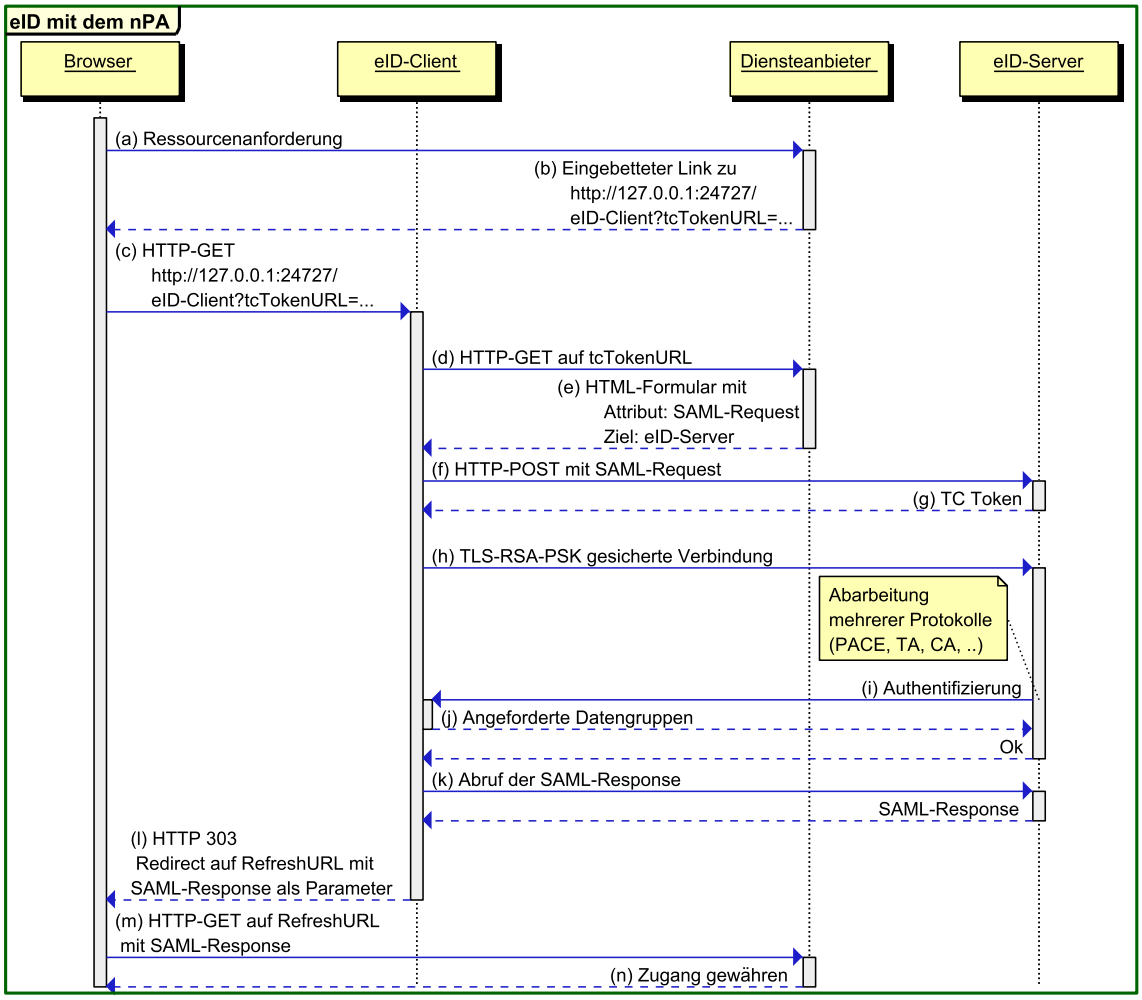

Abbildung 2.9.: Ablauf der eID-Funktion des nPA

(b) Der Dienst leitet den Browser auf die lokale Adresse http://127.0.0.1:24727/eID-Client?tcTokenURL=... um, wobei der Parameter *tcTokenURL* die Adresse angibt, unter der die Clientsoftware die benötigten Parameter für den Authentisierungsvorgang erhält.

(c) Auf dem Rechner des Benutzers läuft ein HTTP-Server, der die Anfrage annimmt und den eigentlichen eID-Client startet.

(d) Der eID-Client fragt die tcTokenURL ab, die auf eine Seite des Diensteanbieters verweisen muss.

(e) Auf dieser Seite befindet sich ein HTML-Formular, das einen SAML-Request[25] ent-

[25] Dabei handelt es sich um einen SAML AuthnRequest, der eine Authentisierungsanfrage für den eID-Server enthält. Siehe dazu [24].

hält.

(f) Das Formular wird automatisch abgeschickt und an den eID-Server geleitet. Der vom Diensteanbieter signierte und für den eID-Server verschlüsselte SAML-Request enthält eine Anfrage zur Authentifizierung des Benutzers. Zusätzlich werden in ihm die benötigten Attributtypen genannt.

(g) Der eID-Server antwortet mit dem TC Token, der unter anderem die benötigten Sessioninformationen und die RefreshURL, unter der der Browser später die SAML-Response abrufen kann, enthält.

(h) Der Dienst baut eine TLS-PSK-RSA gesicherte Verbindung zu der im TC-Token hinterlegten Adresse auf.

(i) Über diese Verbindung findet die eigentliche Authentifizierung mit Hilfe verschiedener Protokolle statt, die in der TR-03112 näher erläutert werden. Währenddessen kann der Benutzer optionale Attribute abwählen und stimmt anschließend dem Vorgang mit seiner eID-PIN zu.

(j) Nachdem die Protokolle erfolgreich durchgelaufen sind, werden die Attribute aus dem nPA ausgelesen und zum eID-Server geleitet.

(k) Der eID-Client ruft die SAML-Response vom eID-Server ab und ...

(l) ... generiert daraus einen HTTP-Redirect auf die RefreshURL mit der SAML-Response als Parameter.

(m) Der Browser ruft die Seite des Diensteanbieters mit der SAML-Response auf.

(n) Diese antwortet mit den angeforderten Ressourcen.

Extended Access Control

Die Extended Access Control (EAC) stellt die Korrektheit des Auslesevorgangs sicher und wird in der TR-03110 [14] spezifiziert. Im Kontext des neuen Personalausweises wird sie durch die drei Protokolle Password Authenticated Connection Establishment (PACE), Terminal Authentication (TA) und Chip Authentication (CA) umgesetzt:

PACE stellt sicher, dass der Benutzer mit dem Auslesevorgang einverstanden ist und stellt den Wissensteil der Authentifizierung dar. Durch die Umsetzung als Challenge-Response-Verfahren wird bei PACE niemals die PIN direkt übertragen. Nach dem erfolgreichen Durchlauf besteht ein Secure-Messaging-Kanal zwischen der Stelle, an dem die PIN eingegeben wurde und dem nPA.

Bei der **TA** authentifiziert sich die auslesende Partei (das Terminal) gegenüber dem nPA. Das Protokoll benutzt asymmetrische Schlüssel, wobei das Wurzelzertifikat (Country Verifying Certificate Authority Zertifikat) im nPA gespeichert ist.

Bei der **CA** authentifiziert sich der nPA gegenüber dem Terminal. Somit ist sichergestellt, dass die ausgelesenen Daten authentisch sind. Am Ende der CA besteht ein kryptografisch gesicherter Kanal zwischen nPA und Terminal, der auch gegen Man-in-the-Middle Angriffe abgesichert ist.

Daten

Die elektronisch auslesbaren Daten (im nPA-Kontext auch Datengruppen genannt) können in statische und dynamische Daten unterteilt werden. Welche Daten ein Terminal auslesen darf, ist im CV-Zertifikat kodiert, das der nPA vom Terminal erhält. Bis auf die Fingerabdrücke stehen alle *statischen Daten* ebenfalls auf der Vorder- und Rückseite des nPAs, wie in Abbildung 2.10 zu sehen ist.

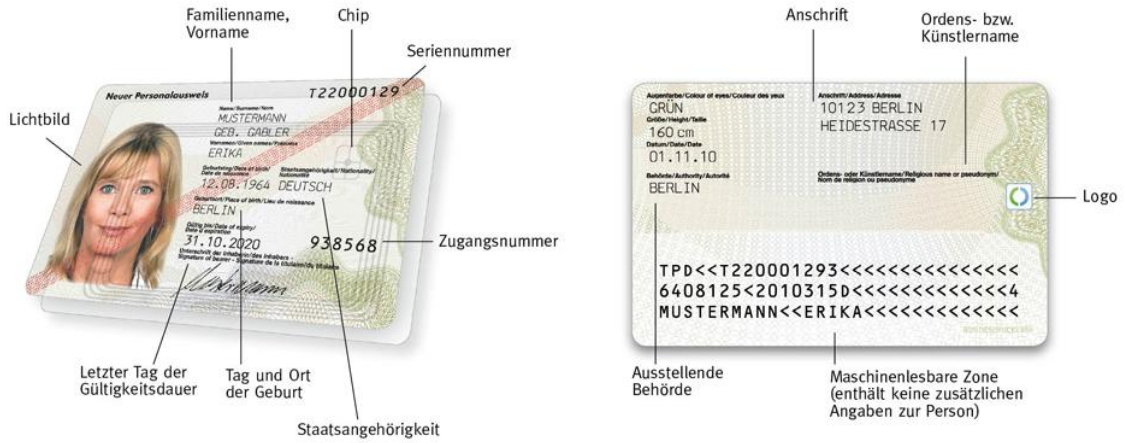

Abbildung 2.10.: Vorder- und Rückseite des neuen Personalausweises - Quelle: [34]

Dynamische Daten werden vom nPA bei jeder Abfrage neu berechnet. Ein Beispiel ist die *Age Verification*, die eine Altersabfrage realisiert. Dabei antwortet der nPA auf die Anfrage, ob das Alter der Person über einem bestimmten Wert liegt, mit einem boolschen Wert. Das konkrete Alter selber wird nicht herausgegeben. Ein weiteres Beispiel ist die Restricted Identification.

Restricted Identification In bestimmten Szenarien reicht es aus, eine Person einer früheren Authentifizierung zuordnen zu können. Dies ist zum Beispiel bei der Realisierung eines Logins oder der Verhinderung von mehrfachen Stimmabgaben bei Onlineumfragen nötig. Für diese Fälle steht der Restricted Identifier (RI) als eindeutig identifizierbares Attribut zur Verfügung, welches zusätzlich sektorübergreifend nicht verlinkbar ist. Das bedeutet, dass ein Diensteanbieter zwar einen Benutzer sitzungsübergreifend wieder erkennen kann,

jedoch ist es für zwei zusammen arbeitende Diensteanbieter unmöglich diesen Nutzer untereinander zuzuordnen. Ermöglicht wird das, indem der RI bei jeder Abfrage dynamisch generiert wird. Der Chip bekommt als Eingabe den öffentlichen Schlüssel PK_{Sector} des Sektors und die Domainparameter D. Zusammen mit seinem privaten Schlüssel SK_{ID} führt er einen Diffie-Hellman-Schlüsselaustausch durch, hasht das Ergebnis und erhält den Restricted Identifier I_{ID}^{Sector}, der zurück zum Terminal geschickt wird.

Kartenleser

In der TR-03119 [21] werden unterschiedliche Kartenleserklassen spezifiziert, die in Abbildung 2.11 zu sehen sind. Der Basisleser verfügt weder über ein Display noch über eine Tastatur und ist nur für die eID-Funktion des nPA gedacht. Standard- und Komfortleser besitzen hingegen Tastatur und Display. Der Komfortleser besitzt zusätzlich ein eigenes Berechtigungszertifikat zur Erstellung der qualifizierten elektronischen Signatur mit dem nPA, die hier nicht weiter betrachtet wird.

Abbildung 2.11.: Komfort-, Standard- und Basislesegerät - Quelle: [35]

Grenzen des Systems

Dem System neuer Personalausweis sind vor allem im mobilen Bereich einige Grenzen gesetzt. Die offiziell angebotene Clientsoftware zum Auslesen des nPA, nämlich die AusweisApp [2], wird bisher nur für Windows und und verschiedene Linux-Distributionen angeboten. Da eine Portierung der über 100 Megabyte großen Anwendung zu aufwändig erscheint, hat Kristian Beilke bereits Ende 2010 in seiner Diplomarbeit [26] über eine

Einschränkung der eCard-API und somit der darauf basierenden Clientsoftware nachgedacht. Ebenfalls wurde der Prototyp einer abgespeckten AusweisApp bereits mehrmals auf der Cebit demonstriert und ist mittlerweile frei verfügbar[26]. Das größere Problem stellt die eingeschränkte Hardware der Smartphones dar. Auch wenn NFC und die vom nPA genutzte ISO-14443 Schnittstelle viele Gemeinsamkeiten haben, treten in der Praxis verschiedene Probleme auf. Während der EAC muss das Terminal an einer Stelle eine extended Length APDU an den nPA senden, welches von den bestehenden NFC-Controllern nicht unterstützt wird. Des Weiteren nimmt der nPA mehr Energie auf, als die NFC-Controller über das generierte Feld zur Verfügung stellen können. Da die Endgerätehersteller bei dem maximalen Energieverbrauch der NFC-Controller sehr enge Grenzen festlegen ist es meiner Meinung nach unwahrscheinlich, dass kurzfristig gesehen ein mobiles Endgerät den nPA direkt auslesen kann.

2.3.2. Mobile Authentifizierung

Die Authentifizierung mit und gegenüber mobilen Endgeräten wird bisher trotz der zunehmenden Verbreitung mobiler Endgeräte kaum betrachtet. Am meisten verbreitet ist immer noch die Eingabe eines Passworts oder einer PIN, sei es um das Telefon zu entsperren oder um sich auf einer Webseite anzumelden. Eine Authentifizierung mit Biometrie, z. B. einem Fingerabdruck oder per Gesichtserkennung, wird bisher nur zur Anmeldung am Betriebssystem genutzt und hat prinzipbedingt Nachteile. Biometrische Merkmale können relativ leicht von einem Angreifer erfasst werden. Trinkt man z. B. ein Glas Wasser, hinterlässt man seine Fingerabdrücke auf diesem. Mit den so erlangten Daten kann ein Angreifer eine Attrappe bauen, um sich gegenüber einem biometrischen System als legitimer Benutzer auszugeben. Sollte der Benutzer von dieser Kopie erfahren, kann er jedoch nicht seine biometrischen Merkmale willentlich ändern. In dem Fall müsste dann das System so verbessert werden, dass es die Attrappe von dem Original unterscheiden kann.

Eine Alternative stellt die Zwei-Faktor-Authentifizierung mit Besitz und Wissen dar, wie sie zum Beispiel bei der eID-Funktion des nPA implementiert ist ist. Der Benutzer besitzt den nPA und weiß die PIN, damit er ihn benutzen kann. Eine wichtige Eigenschaft des Besitzes ist dabei, dass er im Gegensatz zum Wissen nicht oder nur mit erheblichem Aufwand vervielfältigt werden kann. Leitet man Identitätstoken aus einer Datenquelle ab und speichert sie im Dateisystem eines mobilen Endgeräts, fällt diese hingegen Eigenschaft weg. Bei einem Identitätstoken sowie den zugehörigen privaten Komponenten handelt es sich um Software, die z. B. von einem Schadprogramm kopiert werden kann. Die Speicherung eines Teils der privaten Komponenten in einem Secure Element bietet hier einen Ausweg. Das SE bietet ausreichenden Schutz gegen das Auslesen und Kopieren der in ihm gespeicherten Daten und stellt somit den Besitz dar [27]. Um sich authentifizieren zu können, benötigt

[26] `http://sar.informatik.hu-berlin.de/BeID-lab/eIDClientCore`

[27] Anmerkend sei gesagt, dass ein SE in der Regel dauerhaft mit dem mobilen Endgerät verbunden ist und somit ähnliche Sicherheitsbedenken auftreten, wie bei einem dauerhaft auf einem Basisleser aufgelegten nPA [36].

der Benutzer außerdem noch eine PIN, mit dem er das SE freischaltet. Eine solche Lösung wird in Kapitel 4 vorgestellt.

Grundlage dafür ist allerdings, dass sich die privaten Komponenten entsprechend aufteilen lassen. Da der geschützte private Teil nie das SE verlassen darf, muss ein Teil des Authentifizierungsprotokolls im SE durchgeführt werden. Das Format der Identitätstoken sowie die zugehörigen Protokolle werden in sogenannten Anonymous Credential Systemen festgelegt. Im Folgenden werden zwei solcher Systeme vorgestellt und auf ihre Tauglichkeit für die vorliegende Arbeit untersucht.

2.3.3. Anonymous Credential Systeme

Anonymous Credential Systeme (ACS) sind eID-Systeme, die besonderen Datenschutzanforderungen genügen. In einem ACS besitzt ein Benutzer Credentials oder auch Softtoken genannt, die er zur Authentisierung verwendet. In den Token können auch Attribute kodiert sein, die der Benutzer bei der Authentifizierung entweder dem Diensteanbieter präsentieren oder vor ihm verbergen kann. Unabhängig davon welche Attribute aufgedeckt werden, kann der Diensteanbieter verifizieren, dass die Attribute von einer vertrauenswürdigen Instanz ausgestellt wurden. Die Ausstellung eines Tokens wird von dessen Nutzung entkoppelt. Somit weiß die ausstellende Partei (der Tokenissuer) nicht, bei welchem Diensteanbieter sich der Benutzer authentisiert.

Die zwei bekanntesten Vertreter sind der Identity Mixer[28] sowie U-Prove[29].
In der ABC4Trust Projektbeschreibung[8] werden sie (übersetzt) wie folgt beschrieben: „Identity Mixer ist eine Protokollfamilie von der eine Beispielimplementierung existiert. U-Prove ist eine Spezifikation, die die Kernfunktionalität einer größeren Menge von Protokollen bietet.“ Außerdem unterscheiden sie sich in den verwendeten Funktionen, unterstützen Eigenschaften und im Tokenformat und sind somit nicht ohne weiteres interoperabel[30].

Identity Mixer

Der Identity Mixer (IDEMIX) ist ein Anonymous Credential System von IBM, das 2001 erschien und als freie Javaimplementierung verfügbar ist. Es bietet mehr Funktionen als U-Prove ist dafür allerdings auch komplexer. Neben dem bereits erwähnten selektiven Aufdecken von Attributen verfügt es zum Beispiel über eine doppelte Unlinkbarkeit. Damit kann ein Credential[31] weder vom Issuer verfolgt werden, noch ist eine Verknüpfung eines Credentials möglich, das mehrmals präsentiert wird. Des Weiteren können Abfragen mit Hilfe von Gleichheitsbeweisen, Bereichsbeweisen und logischer Verknüpfungen flexibel gestaltet

[28] `http://idemix.wordpress.com`
[29] `http://www.credentica.com`
[30] Dieser Aufgabe hat sich das Projekt ABC4Trust angenommen
[31] In IDEMIX wird ein Token Credential genannt.

werden. Steht zum Beispiel in einem Credential nur das Geburtsdatum und der Geburtsort ist eine Abfrage in der Form „Bist du in den Städten Berlin, Hamburg oder Bremen geboren und bist mindestens 18 Jahre alt?“ möglich.

Auf Grund des Funktionsumfangs ist die Größte Schwäche von IDEMIX die Performanz. Anders als für U-Prove ist für IDEMIX noch keine performante Implementierung vorhanden, die zusammen mit einer SmartCard eingesetzt werden kann. Die bislang einzige Implementierung, die vollständig auf einer SmartCard läuft, benötigt 10 Sekunden zur Präsentation eines Tokens. Da die später präsentierte Lösung jedoch ein Secure Element benötigt, wird in dieser Arbeit nicht näher auf IDEMIX eingegangen.

U-Prove

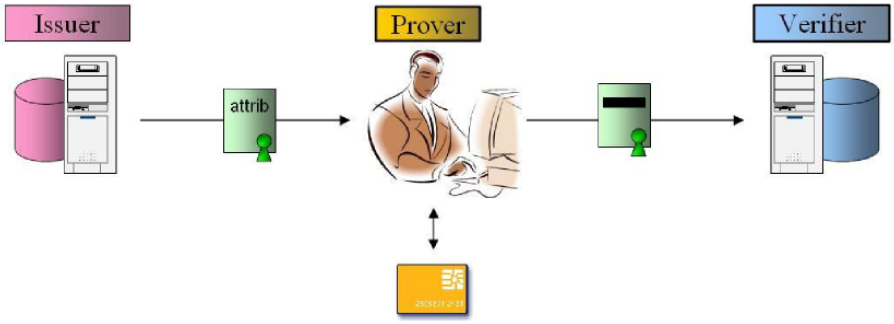

Abbildung 2.12.: Überblick über das U-Prove System - Quelle: [40]

Das U-Prove System wird schematisch in Abbildung 2.12 dargestellt. Kern von U-Prove ist der Identitätstoken, der unter anderem die signierten Attribute enthält. Dieser wird von einer vertrauenswürdigen Instanz, dem **Token Issuer**, für den Benutzer, **Prover** genannt, erstellt. Der Prover kann einen Token benutzen, um sich gegenüber einem Diensteanbieter, **Verifier** genannt, zu authentisieren. Für die Benutzung des Tokens muss er neben dem privaten Schlüssel auch alle kodierten Attribute kennen. U-Prove bietet folgende datenschutzfreundliche Eigenschaften:

Unverfolgbarkeit Bei der Ausstellung bildet der Issuer eine sogenannte blinde Signatur über den Token. Diese hat die Eigenschaft, dass der Issuer sie ebenso wie den öffentlichen Schlüssel des Tokens nicht sieht. Präsentiert der Prover den Token einem Verifier, der mit dem Issuer zusammenarbeitet, können diese den Benutzer nicht über den Token zuordnen. Die einzige Möglichkeit der Zuordnung bietet das Token Information Field, das der Issuer sieht und bei jeder Präsentation aufgedeckt werden muss.

Unverknüpfbarkeit Verifier, die verschiedene Token vom selben Prover präsentiert bekommen, können den Prover nicht zuordnen. Verwendet er den selben Token mehrmals, sind die Benutzungen hingegen verknüpfbar.

Selektive Aufdeckung Bei der Präsentation des Tokens kann der Prover eine Attributteilmenge verdecken und trotzdem die Authentizität und Integrität des Tokens beweisen. Das Token Information Field wird immer aufgedeckt.

Nicht Wiederverwendbarkeit Ein Angreifer, der den Authentifizierungsablauf zwischen Prover und Verifier mitgelesen hat, kann den Mitschnitt nicht dazu verwenden, sich ebenfalls gegenüber dem Verifier zu authentifizieren.

Authentizität, Integrität Die Authentizität und Integrität der Token ist durch die Signatur des Issuers auch dann noch gegeben, wenn einzelne Attribute nicht aufgedeckt werden.

Anonymität, Pseudonymität Verdeckt der Prover bei der Präsentation des Tokens alle Attribute, kann er anonym agieren. Er beweist nur noch, dass der Token von einem vertrauenswürdigen Issuer ausgestellt wird. Durch mehrmalige Verwendung desselben Tokens wird ein Pseudonym realisiert.

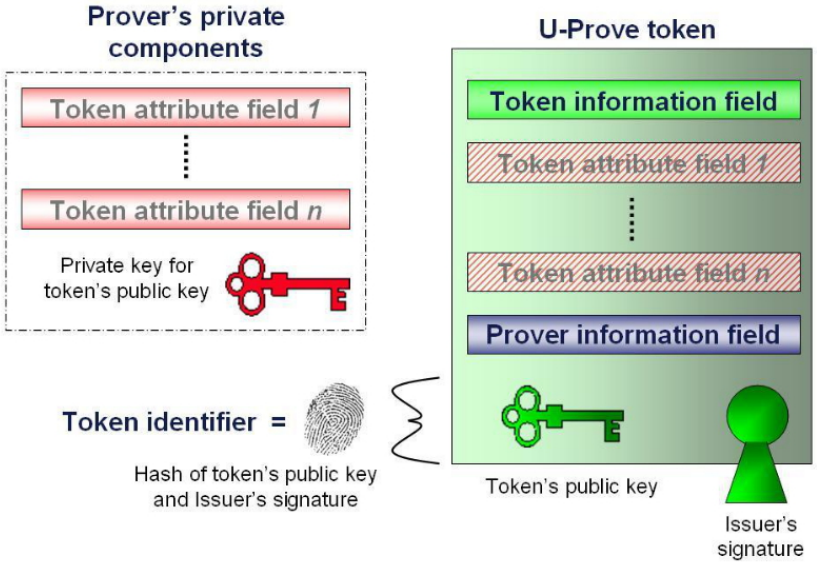

Abbildung 2.13.: Aufbau eines U-Prove Tokens - Quelle: [40]

In Abbildung 2.13 sind die einzelnen Komponenten zu sehen, die beim Benutzer gespeichert werden. Sie teilen sich auf in die privaten Komponenten und den eigentlichen Token. Die privaten Komponenten bestehen aus den verschlüsselten oder im Klartext abgespeicherten Attributen sowie dem privaten Tokenschlüssel. Werden aus einem Datensatz mehrere Token generiert, werden die Attribute in der Regel nur einmal abgespeichert, hingegen besitzt jeder Token ein individuelles Schlüsselpaar. Der eigentliche Token ist wie folgt aufgebaut:

Token Information Field Das Token Information Field enthält einen vom Issuer festgelegten Wert, der immer aufgedeckt wird, wenn der Prover sein Token benutzt. Typischer

Anwendungsfall ist die Anwendung für Metadaten, wie zum Beispiel einen Gültigkeitszeitraum. Wird in dem Feld ein benutzerspezifischer Wert kodiert, kann der Prover darüber verfolgt werden und nicht mehr komplett anonym handeln.

Token Attribute Field Das Token Attribute Field enthält die vom Issuer kodierten Attribute. Beim Benutzen des Tokens können diese Attribute aufgedeckt oder verdeckt werden.

Prover Information Field Das Prover Information Field enthält einen vom Prover festgelegt Wert, der für den Issuer nicht sichtbar ist und bei der Benutzung des Tokens immer aufgedeckt wird.

Token Identifier Der Token Identifier besteht aus dem Hash über den öffentlichen Tokenschlüssel und die Issuer Signatur. Nimmt man eine perfekte Hashfunktion an, kann niemand einen zweiten Token mit dem selben Identifier erzeugen. Der Token Identifier wird bei der Benutzung des Tokens trivialerweise mit übertragen, sodass ein mehrmals verwendeter Token zugeordnet werden kann.

Widerruf Das Widerrufen eines Token ist bei einem ACS durch die Anonymitätseigenschaft generell schwierig. Sofern der Token Identifier bekannt ist, kann die Sperrung über ihn erfolgen. Auch kann der Issuer ein eindeutiges Merkmal in das Token Information Field schreiben, das auf eine Blacklist gesetzt werden kann. Durch das Merkmal entfällt allerdings die Unverfolgbarkeit. Weiterhin können on-demand Token verwendet werden, die erst dann vom Issuer erstellt werden, wenn ein Prover sich authentifizieren möchte. Wird ein Secure Element verwendet, kann der Issuer die Token auf Basis von dessen ID sperren. Auch kann die komplette Authentifizierungsfunktion des Secure Elements durch einen TSM gesperrt werden.

Verwendung eines Secure Elements Nach der U-Prove Spezifikation [39] kann ein Token zusätzlich durch ein spezielles Gerät, im vorliegenden Fall ein Secure Element, geschützt werden. Dafür wird ein Teil des privaten Schlüssels im SE abgelegt. Der Token kann nur mit Hilfe des SE benutzt werden, wobei ein einfaches Challenge-Response Verfahren während der Präsentation zwischen der normalen Applikation im Endgerät und dem SE abläuft. Der Rest des Protokolls wird auf dem Endgerät ausgeführt, da das SE eine zu geringe Rechenleistung bietet. Vorteilhaft ist dabei für den Prover, dass ein Teil des privaten Schlüssels per Hardware geschützt ist und somit nicht von Schadsoftware ausgelesen werden kann.

Vergleich der vorgestellten Systeme

Tabelle 2.1.: Vergleich von IDEMIX, U-Prove und der eID-Funktion des nPA

Eigenschaft	IDEMIX	U-Prove	nPA
Selektives Aufdecken	ja	ja	ja
Unverfolgbarkeit	ja	ja	ja[1]
Unverknüpfbarkeit	ja	ja [2]	ja[1]
Gleichheitsbeweise	ja	nein	nein
Logische Verknüpfungen	ja	nein	nein
Bereichsbeweise	ja	**nein**[7]	ja[6]
Begrenzte Wiederverwendung	ja[3]	ja[3]	nein
Revocation	ja[3]	ja[3]	ja
Authentifizierung des Diensteanbieters	**nein**	**nein**	ja
Ausstellungsdauer	<1s	<1s	≈ 3Wochen[4]
Authentifizierungsdauer	<3s	<2s	>3s
SmartCard-Integration	**nein**[5]	ja	ja
SmartCard-Auth.dauer	≈ 11s	<1s	>5s

[1] Die Unverknüpfbarkeit ist gegenüber verschiedenen Diensteanbietern. Gegenüber dem eID-Server wird sie nur organisatorisch hergestellt.
[2] Für jede Authentifizierung muss ein neuer Token verwendet werden.
[3] Die Eigenschaften sind nicht von Hause aus in den verfügbaren Bibliotheken vorhanden, können aber relativ leicht implementiert werden.
[4] Beinhaltet auch organisatorische Prozesse
[5] Es existiert zwar eine Veröffentlichung dazu, jedoch keine frei verfügbare Implementierung
[6] Nur für das Alter und den Wohnort
[7] Wird von der unterliegenden Kryptografie unterstützt, ist jedoch noch nicht implementiert.

In der Tabelle 2.1 werden die Eigenschaften von U-Prove, IDEMIX und der eID-Funktion des nPA miteinander verglichen. Wie an den vielen Fußnoten der Tabelle erkennbar ist, ist ein Vergleich der drei Systeme schwierig. Zum einen unterscheiden sich bei den ACS die theoretischen Grundlagen von den implementierten Eigenschaften. Zum anderen werden beim System nPA bestimmte Eigenschaften nur organisatorisch sichergestellt. Trotzdem ist ein direkter Vergleich wichtig. Sollen Daten vom nPA abgeleitet und in einem Identitätstoken gespeichert werden, muss das verwendete ACS mindestens dieselben datenschutzfreundlichen Eigenschaften besitzen, wie der nPA.

Dabei fällt auf, dass beide Systeme Einschränkungen besitzen. Für IDEMIX existiert bis heute noch keine performante Implementierung, die mit einem Secure Element verwendet werden kann. Diese Eigenschaften ist jedoch auf Grund der Sicherheitseigenschaften mobiler Betriebssysteme (Abschnitt 2.1.2) für die Lösung unbedingt erforderlich. In U-Prove unterstützt zwar die unterliegende Kryptografie Bereichsbeweise, diese sind allerdings noch

nicht in der offiziellen Implementation umgesetzt. Bei der Benutzung des nPA muss sich der Diensteanbieter mit seinem Berechtigungszertifikat ausweisen, wobei dem Benutzer unter anderem der Name des Dienstes und der Auslesegrund angezeigt werden. Ein Diensteanbieter erhält solch ein Zertifikat von der Vergabestelle für Berechtigungszertifikate. Diese überprüft zuvor, ob der Anbieter nach den gesetzlichen Bestimmungen überhaupt für das Zertifikat berechtigt ist. Solch ein Zwang existiert bei den bisherigen U-Prove und IDEMIX Implementierungen nicht, kann aber hinzugefügt werden.

In der vorliegenden Arbeit entscheide ich mich für eine auf U-Prove basierende Lösung, da die Implementierung eines zusätzlichen Features in U-Prove sehr wahrscheinlich einfacher ist, als eine SE-Integration für IDEMIX zu implementieren.

3. Anforderungen

In den vorherigen Kapiteln wurde ein grober Überblick über für diese Arbeit relevanten Aspekte bei mobilen Endgeräten, Secure Elements und Authentifizierungsmethoden im Internet gegeben. Auf dieser Basis werden im Abschnitt 3.2 Annahmen formuliert, die für das Erreichen der Schutzziele aus Abschnitt 3.1 benötigt werden. Zusätzlich wird auf der Angreiferseite betrachtet, über welche Voraussetzungen der Angreifer verfügt und welche Ziele er verfolgt. Darauf aufbauend werden in Abschnitt 3.4 Bedrohungsszenarien vorgestellt, die bei der Konstruktion der Lösungsvorschläge beachtet werden müssen.

3.1. Schutzbedarf

Es existieren verschiedene *Grundwerte der Informationssicherheit*, welche unter anderem im BSI-Gloassar genannt werden[1]. Die Bedeutung der drei Grundwerte Vertraulichkeit, Integrität und Authentizität für den vorliegenden Anwendungsfall wird im Folgenden dargestellt:

Vertraulichkeit Die Attribute, die aus der Datenquelle ausgelesen werden, sind vertraulich und dürfen während der Übertragung zum mobilen Endgerät und der Speicherung darin nur von den berechtigten Parteien gelesen werden. Das gilt ebenfalls für die U-Prove Token und alle privaten Komponenten des U-Prove Protokolls.

Integrität Die Korrektheit (Unversehrtheit) der im Gerät abgespeicherten Token und der damit verbundenen Anwendungen und Systeme muss sichergestellt sein.

Authentizität Alle beteiligten Akteure müssen verifizierbar authentisch sein. Ebenfalls muss die Authentizität der Attribute und abgeleiteten Token überprüfbar sein.

Der eigentlich ebenfalls wichtige Grundwert *Verfügbarkeit* wird hingegen nicht betrachtet, da dies das Verfahren bei geringem Mehrwert ungleich verkomplizieren würde. So kann zum Beispiel Schadsoftware auf dem Endgerät die Internetverbindung stören oder das Secure Element sperren, ohne dass dies effektiv verhindert werden kann.

[1] `http://www.bsi.bund.de/DE/Themen/weitereThemen/ITGrundschutzKataloge/Inhalt/Glossar/glossar_node.html`

Bei der Speicherung und Übertragung sind eine Reihe von Akteuren involviert, zwischen denen verschiedene Verbindungen aufgebaut werden. Da diese jeweils unterschiedliche Voraussetzungen sowie Sicherheitsanforderungen besitzen, findet deren Sicherheitsbetrachtung in den entsprechenden Unterkapiteln statt.

Schutzziele

Z_1 Bei der Übertragung der Token und privaten Komponenten (Attribute und geheime Schlüssel) muss deren Vertraulichkeit und Integrität gewährleistet sein.

Z_2 Die Token und privaten Komponenten müssen in dem korrekten Gerät gespeichert werden.

Z_3 Es dürfen nur berechtigte Akteure in vorgegebener Weise auf die Token und privaten Komponenten zugreifen.

Bemerkungen

Z_1 Auch wenn bereits Techniken zur sicheren Datenübertragung bestehen, sind diese eventuell nicht direkt auf das Szenario anwendbar. Mögliche Probleme können sich durch die Einbettung des Secure Elements in das Smartphone ergeben, wodurch Angriffe durch Schadsoftware auf dem Smartphone ermöglicht werden. Des Weiteren sollte betrachtet werden, inwiefern SE und TSM bereits Kenntnis voneinander haben müssen, um Man-in-the-middle Angriffe verhindern zu können.

Z_2 Hierbei sollen Angriffe abgewehrt werden, bei denen die Daten des Benutzers in ein vom Angreifer kontrolliertes Gerät umgeleitet werden. Dies ist relativ leicht sicherzustellen, wenn Issuer und Secure Element direkt miteinander verbunden sind. Erfolgt die Kommunikation allerdings über weitere Zwischenstationen (Internetdienstanbieter, Smartphone, ...), müssen zusätzliche Vorkehrungen getroffen werden. Eine zusätzliche Herausforderung ist es, wenn der Issuer keine Zuordnung zwischen Benutzer und SE besitzt. Dabei könnte er im Extremfall nur sicherstellen, dass die Daten an ein authentisches SE übertragen werden, allerdings nicht, ob es dem Benutzer gehört.

Z_3 Die Frage, wer für welche Operationen berechtigt ist, wird in Abschnitt 4.1.2 geklärt.

3.2. Annahmen

A_1 Die Datenquelle kann nur mit Zustimmung des Benutzers ausgelesen werden.

Beim deutschen eID-System muss der nPA mit dem Kartenleser verbunden sein und der Benutzer muss seine PIN eingeben. Geht man davon aus, dass ein Standard- oder Komfortleser verwendet wird, muss die PIN-Eingabe auf dem Kartenleser erfolgen.

A_2 Dem Benutzer wird beim Auslesevorgang aus der Datenquelle auf einem vertrauenswürdigen Display der Auslesegrund bzw. der Name des auslesenden Akteurs angezeigt.

Wird der neue Personalausweis in Verbindung mit einem Standard- oder Komfortleser verwendet, wird der Inhaber des Berechtigungszertifikats auf dem Leserdisplay angezeigt. Ein ähnlicher Ablauf muss auch bei alternativen Datenquellen umgesetzt werden.

A_3 Das Betriebssystem des SE wird als sicher angesehen.

Chipkartenbetriebssysteme sind in der Regel sicherheitsevaluiert und weniger komplex als Rich Execution Environments[2]. Des Weiteren kommunizieren sie nach außen über ein eingeschränktes APDU-Interface, welches eine geringe Angriffsfläche bietet. Auf der Annahme bauen die folgenden zwei Punkte auf.

A_4 Nur berechtigte Parteien besitzen administrative Rechte auf dem SE.

Im GlobalPlatform Standard wird festgelegt, welche Parteien welche Aktionen auf dem Secure Element ausführen dürfen. Es wird davon ausgegangen, dass diese Zugriffsmechanismen nicht umgangen werden können.

A_5 Java-Card-Applets sind im SE voneinander abgeschottet.

Die Abschottung findet in einem SE auf mehreren Ebenen statt. Die im GlobalPlatform Standard beschriebene Security Domain ist für die Verwaltung von Schlüsselmaterial zuständig. Nur Applets innerhalb derselben SD haben Zugriff auf denselben Schlüssel. Die Applet-Firewall von JavaCard beschränkt die Kommunikation zwischen Applets. In der Praxis müssen die Anforderungen vom Betriebssystem, wie zum Beispiel JCop, umgesetzt werden. Da diese in der Regel sicherheitsevaluiert sind, wird davon ausgegangen, dass sich keine schwerwiegenden Lücken darin befinden, die unberechtigten Zugriff von einem Applet auf ein anderes ermöglichen.

A_6 Der verwendete Kartenleser (Standard oder Komfort) ist nicht manipuliert und besitzt eine „sichere“ Anzeige und Eingabe.

Ein nach der TR-03119 zertifizierter Standard- bzw. Komfortleser bietet neben einem Display, auf dem Daten angezeigt werden, die unverändert zur Karte weitergeleitet werden, eine Tastatur zur PIN-Eingabe, sodass die PIN nicht von einem Angreifer ausgelesen werden kann. Es wird davon ausgegangen, dass der Kartenleser nicht von einem Angreifer manipuliert wurde.

A_7 Token Issuer, ISD-Inhaber und TSM sind vertrauenswürdig.

Jeder der genannten Akteure spielt eine wichtige Rolle, sodass bei Missbrauch oder Manipulation die Sicherheit des Systems gefährdet wäre. Hinzu käme ein Vertrauensverlust auf Benutzerseite. Aus diesen Gründen wird davon ausgegangen, dass die

[2] Dabei handelt es sich um Betriebssysteme, wie zum Beispiel Windows oder Android, die auf typischen Consumergeräten wie Smartphones oder PCs laufen.

genannten Akteure gutartig sind und Sicherheitsmaßnahmen entwickelt haben, welche eine Manipulation sehr unwahrscheinlich werden lassen.

A_8 Die Kommunikation zwischen TSM und ISD-Inhaber und zwischen TSM und JC-Applet-Hersteller ist vertraulich und integer. Außerdem haben die genannten Akteure ein Vertragsverhältnis etabliert.

Der Applet-Hersteller installiert das Applet nicht selbst im SE sondern überlässt dies dem TSM, wobei die Kommandos dafür vom SE-Inhaber erstellt werden. Somit muss das Applet irgendwann vom Applet-Hersteller über den TSM zum SE-Inhaber geleitet werden. Es wird davon ausgegangen, dass die Vertraulichkeit und Integrität des Applets während der Übertragung gegeben ist.

A_9 Die Attribute gelangen vertraulich und integer von der Datenquelle zum Token Issuer.

Geht man davon aus, dass es sich bei der Datenquelle um ein ähnlich abgesichertes Dokument handelt, wie der neue Personalausweis, so ist die Annahme durchaus realistisch. Wird die Verbindung zu dem korrekten Token Issuer aufgebaut, ist die Vertraulichkeit und Integrität durch EAC gewährleistet. Problematisch wird es, wenn der Benutzer auf eine gefälschte Seite geleitet wird, die auf die Datenquelle zugreifen will. Jedoch kann davon ausgegangen werden, dass solch bösartige Diensteanbieter kein Berechtigungszertifikat bekommen.

3.3. Angreifermodell

Um die Anforderungen an die Datensicherheit erfüllen zu können, muss betrachtet werden, welche Möglichkeiten ein Angreifer besitzt und welche Ziele er verfolgt. Grundsätzlich wird davon ausgegangen, dass die beteiligten Akteure, also Benutzer, Token Issuer, Diensteanbieter und eventuelle weitere vertrauenswürdige Instanzen keine bösen Absichten haben und sich gegenseitig vertrauen. Sollte der Token Issuer oder Diensteanbieter doch böswillig sein, können Angriffe von ihnen teilweise durch die U-Prove Protokolle abgewehrt werden[3], was jedoch nicht Teil dieser Arbeit ist. Die mutwillige Weitergabe von privaten Informationen an Dritte kann nicht verhindert werden und wird deswegen ebenfalls ausgeschlossen. Auch wenn der Benutzer gutartig ist, können seine verwendeten Geräte manipuliert und mit Schadsoftware versehen sein.

3.3.1. Malware

Ein Angreifer hat Schadsoftware (Malware) mit Administrationsrechten auf dem PC oder mobilen Endgerät des Benutzers platziert. Mit Hilfe der Malware kann er, wie in Abbildung 3.1 zu sehen ist, Einfluss auf die Benutzergeräte nehmen.

Der Angreifer besitzt zusammengefasst folgende Möglichkeiten:

[3] Siehe dazu [40].

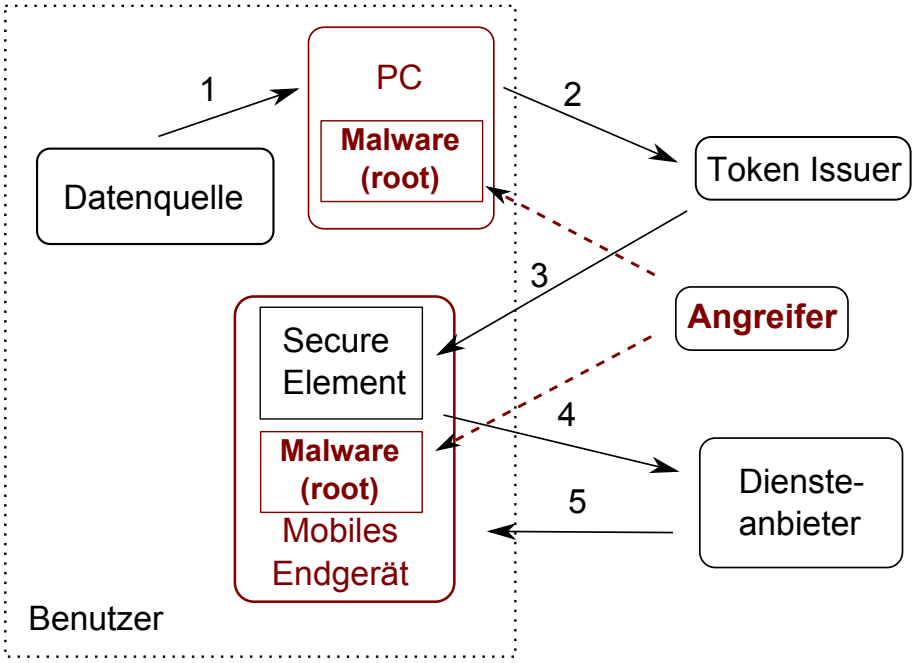

Abbildung 3.1.: Externer Angreifer

- Er besitzt root-Rechte auf dem mobilen Endgerät und dem PC des Benutzers.

  Da die Malware root-Rechte besitzt, kann der Angreifer prinzipiell alle Aktionen auf dem Gerät ausführen, die keinen physischen Zugriff benötigen. So kann er zum Beispiel eingegebene und ausgegebene Daten einsehen und beeinflussen und lesend sowie schreibend auf den Fest- und Arbeitsspeicher des Gerätes zugreifen. Insbesondere kann er über die Malware auch als Man-in-the-Middle agieren, weswegen diese Angreiferklasse nicht extra erwähnt wird.

- Er besitzt keinen physischen Zugriff auf die Geräte des Benutzers, insbesondere auf den PC, das mobile Endgerät und das SE.

  Der Angreifer benötigt Malware, die auf die Funktionstüchtigkeit des Betriebssystems und Endgerätes angewiesen ist. Ist das Endgerät zum Beispiel ausgeschaltet oder besitzt keinen Netzempfang, kann der Angreifer damit nicht interagieren.

- Seine Ressourcen sind beschränkt.

  Er verfügt über eine begrenzte Anzahl an Computern und Zeit. Außerdem besitzt er kein geheimes Wissen.

- Er kann das mobile Endgerät und den PC des Benutzers einander zuordnen.

  Hat der Angreifer Zugriff auf eine große Anzahl an Geräten, kann es schwierig für ihn sein, zwei davon einem bestimmten Benutzer zuzuordnen. Nichtsdestotrotz kann

er durch das Vergleichen von Benutzernamen oder Testen, ob sich Geräte zu einem Zeitpunkt im selben LAN befinden, die Wahrscheinlichkeit für das Finden solcher Paare erhöhen.

Ziel des Angreifers ist es zum einen, jedwede Art an Information über den Benutzer zu erlangen. Hauptsächlich möchte er in den Besitz der Token, sowie der privaten Komponenten kommen, damit er sich als der Benutzer ausgeben kann. Ein weiteres Ziel ist es, die Authentifizierungsfunktion auf dem Endgerät zu nutzen, ohne dass es der Benutzer merkt.

3.4. Bedrohungsszenarien

In diesem Abschnitt werden verschiedene allgemeine Bedrohungen genannt. Auf Grund der in Abschnitt 3.2 genannten Annahmen sind dabei nur Bedrohungen von Bedeutung, die ein Benutzergerät betreffen. Angriffe auf Issuer, Diensteanbieter und eventuelle weitere vertrauenswürdige Instanzen sind nicht Teil dieser Arbeit und deswegen von der Bedrohungsanalyse ausgeschlossen. Des Weiteren werden auf Grund der in 3.1 genannten Gründe keine Angriffe auf die Verfügbarkeit, wie zum Beispiel durch Distributed Denial of Service Attacken, betrachtet.

3.4.1. Kopieren der Attribute

Der Angreifer gelangt in den Besitz der unverschlüsselten Benutzerattribute. Auch wenn er sich in dem gewählten Szenario damit nicht authentisieren kann, da er die zugehörigen U-Prove-Token nicht besitzt, existieren eine Reihe von Diensteanbietern, denen die reinen Benutzerangaben zur Authentifizierung reichen. Zusätzlich spielt es eine Rolle, welche Attribute gestohlen wurden. So kann zum Beispiel davon ausgegangen werden, dass der Verlust einer restricted ID weniger schwerwiegend ist, als der einer Kreditkartennummer.

3.4.2. Kopieren eines Identitätstoken

Der Angreifer gelangt in den Besitz eines oder mehrerer unverschlüsselter Identitätstoken. Verwendet der Benutzer diese Token zur Authentifizierung, kann der Angreifer ihn über die Tokensignatur verfolgen, sodass die Anonymität, die U-Prove eigentlich bietet, ausgehebelt wird. Auf Grund der fehlenden privaten Informationen nützen die Token dem Angreifer hingegen nicht zur eigenen Authentifizierung.

3.4.3. Identitätsdiebstahl

Diese Bedrohung kombiniert die beiden vorherigen. Der Angreifer ist in den Besitz eines U-Prove Tokens sowie der dazugehörigen privaten Informationen gelangt und kann sich somit als Benutzer ausgeben. Dies kann entweder geschehen, indem er die im Benutzergerät gespeicherten Daten kopiert oder so in das Issuing der U-Prove Token eingreift, dass die Token sowie privaten Informationen in ein von ihm kontrolliertes Gerät gelangen.

3.4.4. Missbrauch der mobilen eID-Funktion

Das Endgerät ist bereits eingerichtet, sodass sich der Benutzer damit authentifizieren kann. Mit verschiedenen Angriffsvarianten (Malware, Man-in-the-Middle) kann ein Angreifer die mobile eID des Benutzers zur Authentifizierung benutzen, auch wenn er es nicht schafft, die Daten auf ein eigenes Gerät zu kopieren. Der Angreifer muss somit über eine funktionierende Verbindung zum Benutzergerät verfügen. Im Gegensatz zu den anderen Bedrohungen tritt diese erst nach dem Issuing ein und gehört somit eigentlich nicht mehr zum Thema der Diplomarbeit. Jedoch beeinflussen der Ort und die Zugriffsbeschränkungen der gespeicherten Benutzerdaten stark, wie leicht ein solcher Angriff durchführbar ist.

3.4.5. Klassifikation von Angriffen

Um die Angriffe besser einschätzen zu können, werden sie nach zwei Kriterien bewertet. Das erste Kriterium gibt an, wie schwerwiegend die Folgen des Angriffes einzuschätzen sind. Je mehr Sterne ein Angriff besitzt, um so schwerwiegender sind seine Folgen. Die Aufteilung wurde dabei nach selbst festgelegten Regeln vorgenommen.

leicht (*) Der Angriff führt nur mittelbar zu einem der beschriebenen Bedrohungsszenarien. Abwehrmaßnahmen sind zwar wünschenswert, jedoch nicht zwingend erforderlich.

mittel (**) Der Angriff dient als Vorbereitung für ein Bedrohungsszenario, wobei jedoch noch weitere Voraussetzungen erfüllt werden müssen. Abwehrmaßnahmen sollten vorhanden sein. Ausnahmen müssen begründet werden.

schwer (***) Durch den Angriff tritt unmittelbar eines der Bedrohungsszenarien ein. Abwehrmaßnahmen müssen vorhanden sein.

Das zweite Kriterium gibt an, wie gut der Angriff durch Gegenmaßnahmen verhindert werden kann. Sofern ein Angriff mit drei Sternen nicht verhindert werden kann, wird dieser zusätzlich mit einem grafischen Ausrufezeichen gekennzeichnet.

Wird verhindert Gegenmaßnahmen sind bereits Stand der Technik und können implementiert werden. Der Angriff wird somit vollständig verhindert.

Verhinderbar Es existieren zwar Gegenmaßnahmen, diese werden allerdings noch nicht im breiten Umfeld eingesetzt und sind eventuell auch nur als Prototyp vorhanden. Eine relevante Marktdurchdringung ist meiner Meinung nach hingegen innerhalb der nächsten Zeit realistisch. Ein Beispiel dafür ist die Trusted Execution Environment, die bisher nur im Samsung Galaxy S3 vollständig implementiert ist, wobei mit weiteren Implementierungen in nächster Zeit zu rechnen ist.

Wird nicht verhindert Mit gegenwärtigen Mitteln sind keine Gegenmaßnahmen denkbar. Ein Angriff wird ebenfalls so klassifiziert, wenn nur Gegenmaßnahmen gegen bestimmte Angriffsvarianten existieren oder nur mit unrealistisch hohem Aufwand umsetzbar sind.

4. Speicherung der Identitätstoken

Nachdem die vorherigen Kapiteln mobile Endgeräte und Anonyme Credentials betrachteten, wird nun ermittelt, wie Identitätstoken sicher in einem mobilen Endgerät gespeichert werden können. Zuerst werden die in Kapitel 3 genannten Anforderungen konkretisiert und es wird betrachtet, inwiefern diese die Speicherung beeinflussen. Auf dieser Basis werden verschiedene Lösungsvorschläge vorgestellt und anschließend einer informellen Sicherheitsbetrachtung unterzogen. Steht der Speicherort der Identitätstoken fest, kann im anschließenden Kapitel 5 die Übertragung der Token zu diesem betrachtet werden.

4.1. Zusätzliche Anforderungen

4.1.1. Zu speichernde Daten

Unabhängig vom Speicherort handelt es sich bei den Token um U-Prove Token, sodass folgende Daten zu speichern sind:

Private Komponenten Zu den privaten Komponenten gehören die im Token kodierten *Attribute* und der jeweils zugehörige *private Schlüssel*. Diese werden für das Proving-Protokoll und somit für den Beweis der Tokengültigkeit benötigt. Gelangt ein Angreifer in deren Besitz, kann er sich ohne den dazugehörigen Token zwar nicht authentifizieren, jedoch können die Attribute auch sensible Informationen, wie etwa eine Kreditkartennummer, enthalten, sodass sie vor unberechtigtem Zugriff geschützt werden müssen.

Eigentlicher Token Er bestehet aus *Metainformationen*, *kodierten Attributen*, *Signatur* und dem *öffentlichen Tokenschlüssel*. Ein Token kann zwar ohne Kenntnis des privaten Schlüssels und der verarbeiteten Attribute nicht verwendet werden, trotzdem sollte er vor unbefugtem Lesezugriff geschützt werden, da zumindest die Metainformationen unverschlüsselt vorliegen. Des Weiteren kann ein Angreifer den Token Identifier dazu benutzen, um den Benutzer zu verfolgen.

Programm Auch wenn es sich bei einem Programm nicht um Daten handelt, muss darauf geachtet werden, in welcher Umgebung es gespeichert und ausgeführt wird. Der Schutz der privaten Komponenten sowie der Token muss auch während der Erstellung sowie der Präsentation gegeben sein. Somit müssen die sicherheitskritischen Stellen des Programms gegen Manipulation und sensible Daten während der Ausführung vor Auslesen aus dem Arbeitsspeicher geschützt sein.

4.1.2. Zugriffsrechte

Im vorherigen Abschnitt wurde betrachtet, welche Daten zu speichern sind und ein kurzer Überblick über die Zugriffsbeschränkungen gegeben. Im Folgenden sollen die Zugriffsrechte für die Daten genauer spezifiziert werden, wobei zwischen den drei Akteuren Benutzer, Issuer und Diensteanbieter unterschieden wird. Die Berechtigungen teilen sich bei den Daten in Lese-, Schreib- und Löschberechtigung und bei der Anwendung in Installier-, Ausführ-, Lösch- und Sperr- sowie Entsperrberechtigungen auf. Modifikationen von Daten können als Löschen mit nachträglichem Schreiben interpretiert werden und müssen so nicht extra aufgeführt werden. Mit der Lese- und Schreibberechtigung ist nicht der direkte Zugriff auf den Speicher, sondern vielmehr der Datenfluss gemeint, sodass der Diensteanbieter zum Beispiel eine Leseberechtigung für die Token besitzt, diese jedoch nie selber aus dem Endgerätespeicher ausliest. Grundsätzlich soll dem Nutzer die volle Kontrolle über seine eigenen Daten gegeben werden, sodass dieser fast alle Berechtigungen besitzt. Des Weiteren geschieht die Betrachtung der Operationen kontextfrei, was bedeutet, dass die Berechtigungen voneinander und vom Anwendungsfall unabhängig betrachtet werden. Probleme, die dadurch entstehen und weitere Einschränkungen bedingen, werden in späteren Abschnitten besprochen. Wichtig ist noch zu erwähnen, dass alle anderen Zugriffsversuche blockiert werden müssen.

Private Komponenten

Tabelle 4.1.: Berechtigungen für die Attribute

	Benutzer	Issuer	Diensteanbieter
Lesen	ja	nein*	ja*
Schreiben	nicht anwendbar	ja	nein
Löschen	ja	nein	nein

* Mit Einschränkungen

Bei den privaten Schlüsseln, die ebenfalls zu den privaten Komponenten gehören, handelt es sich um sehr sensible Informationen, sodass keiner der Akteure Zugriff darauf haben darf. Idealerweise werden sie in einem Secure Element erzeugt und verlassen dieses niemals. Somit sind die Schlüssel selbst dann noch sicher, wenn der Benutzer Administrationsrechte auf seinem Gerät besitzt und somit den Anwendungsspeicher auslesen kann. Auf seine Attribute besitzt der Benutzer grundsätzlich alle Zugriffsrechte, wobei der Schreibzugriff nur für das Tokenissuing benötigt wird. Ändert oder löscht man die privaten Schlüssel oder die Attribute, hat das zur Folge, dass die Token nicht mehr verwendet werden können. Zwar bekommen Issuer und Diensteanbieter während des Issuings bzw. während der Präsentation einen Einblick in eine Teilmenge der Attribute, die Auswahl der Attribute wird jedoch vom Benutzer bestimmt.

Token

Tabelle 4.2.: Berechtigungen für Token

	Benutzer	Issuer	Diensteanbieter
Lesen	ja	nein	ja*
Schreiben	nicht anwendbar	ja*	nein
Löschen	ja	nein	nein

* Mit Einschränkungen

Die Berechtigungen für die Token sind ähnlich gelagert, wie diejenigen für die Attribute. Auch hier hat der Benutzer wieder sämtliche Zugriffsrechte. Der Issuer erstellt während des Issuingprozesses zusammen mit dem Benutzer die Token. Für das endgültige Schreiben ist jedoch die Anwendung des Benutzers zuständig. Das Lesen der Token läuft ähnlich ab. Auch hier erhält der Diensteanbieter während der Präsentation nur das Token, welches ihm von der Clientseite zur Verfügung gestellt wird. Da die Sperrung von Token im Hintergrundsystem geschehen muss, wird sie hier nicht betrachtet.

Applikation

Tabelle 4.3.: Berechtigungen für Anwendung

	Benutzer	Issuer / TSM	Diensteanbieter
Installieren	Initiierung	Durchführung	nein
Ausführen	ja	nein	nein
Löschen	Initiierung	Durchführung	nein
Sperren	ja	ja	nein
Entsperren	ja[1]	ja	nein

[1] Sofern nicht vom Issuer / TSM gesperrt

Die Berechtigungen für die Applikation unterscheiden sich etwas von den beiden vorherigen Fällen. Unter dem Sperren von Applikationen werden Mechanismen verstanden, die deren Ausführung verhindern. Anders als beim Löschen sind die Applikationen jedoch noch im Endgerät vorhanden und können durch ein Entsperrkommando wieder zugänglich gemacht werden. Dies kann vom Benutzer unter anderem dazu verwendet werden, um eine ungewollte Benutzung der Token zu verhindern. Zusätzlich hat der Applikationshersteller (in der Regel ist das der Issuer) die Möglichkeit diese auf allen Geräten zu sperren, sofern schwerwiegende Fehler in der Applikation gefunden wurden. Auch ist denkbar, dass Programme nur auf bestimmten, eventuell kompromittierten, Geräten gesperrt werden. Um einen Missbrauch des Issuers zu verhindern, sollte die Sperrung nur über den TSM geschehen können.

Bei der Benutzung eines Secure Elements ist darauf zu achten, dass der Benutzer für die Installation und Löschung des Applets den TSM benötigt. Die Sperrung und Entsperrung kann ohne Interaktion mit dem TSM über eine PIN erfolgen.

Probleme der Nutzerzentrierung

Die weitreichenden Berechtigungen des Benutzers können missbraucht werden. Gelingt es ihm, die privaten Schlüssel der Token ebenfalls auszulesen, kann er die Token kopieren und auf anderen Geräten verwenden. Somit kann er seine digitale Identität an andere Personen weitergeben oder versuchen ein Token mehrmals zu verwenden[1], wobei nur der zweite Fall durch U-Prove Mechanismen verhindert werden kann. Befindet sich Malware auf dem Endgerät, die sich als der Benutzer ausgibt, kann sie versuchen die elektronische Identität des Benutzers zu stehlen, wobei bereits „nur“ der Diebstahl von Attributen oder Token kritisch wäre, sofern diese nicht extra verschlüsselt sind. In jedem Fall muss solch unberechtigtes Kopieren unterbunden werden.

4.2. Speicherung im Dateisystem

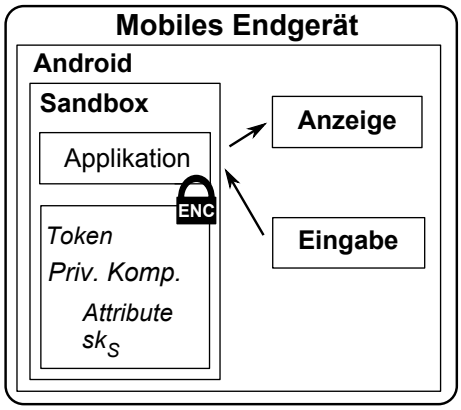

Abbildung 4.1.: Speicherung im mobilen Endgerät

Die Speicherung der Identitätstoken im Endgerätespeicher wird schematisch in Abbildung 4.1 dargestellt. Auch wenn in Abschnitt 2.1.2 bereits Gründe aufgezeigt wurden, wieso diese Lösung zu unsicher ist und deswegen insbesondere Anforderung III aus Abschnitt 3.1 nicht erfüllt werden kann, soll sie der Vollständigkeit halber trotzdem kurz betrachtet werden.

Bei der *Applikation* handelt es sich um eine Android-App, die in einer eigenen virtuellen Maschine und somit einer eigenen Sandbox ausgeführt wird. Mit ihr kann sich der Benutzer

[1] Ein ähnliches Problem stellt das Cardsharing beim Pay-TV dar. Dabei wird eine Dekoderkarte zur gleichzeitigen Entschlüsselung mehrerer Videostreams verwendet.

neue Token auf das Endgerät laden und sich anschließend authentifizieren. Im privaten Speicher sind die *Token* so wie die privaten Komponenten, die aus den *Attributen* und dem privaten Schlüssel sk_s bestehen, verschlüsselt abgespeichert. Die Verschlüsselung basiert auf einem symmetrischer Schlüssel, der durch ein vom Benutzer festgelegtes Passwort geschützt ist.

Vorteile Gegenüber einem Secure Element stehen dem Endgerät mehr Ressourcen zur Verfügung. Somit ist davon auszugehen, dass die Rechenkapazität und der Speicher für eine performante Implementierung des U-Prove Protokolls ausreichen. Des Weiteren verfügt es über einen Monitor, auf dem Tokeninhalte und Transaktionsinformationen zur Kontrolle angezeigt werden können.

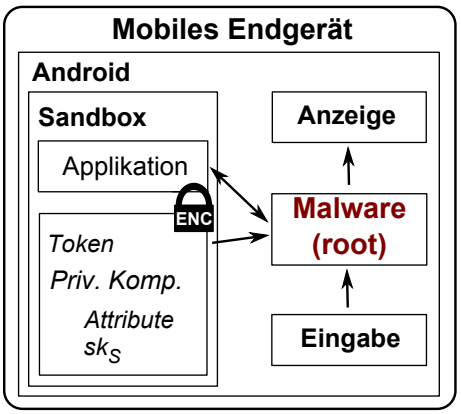

Abbildung 4.2.: Angriff mobiles Endgerät

Probleme Die ist Lösung zu unsicher. Ein Angreifer kann mit Schadsoftware, wie in Abbildung 4.2 zu sehen ist, die gespeicherten verschlüsselten Daten auslesen und auf seinen PC übertragen, ohne dass der Benutzer Kenntnis davon erlangt. Zusätzlich kann er die Eingaben mitschneiden. Möchte der Benutzer nun irgendwann seine Token benutzen und gibt sein Passwort ein, kann der Angreifer es abgreifen und somit anschließend die Token verwenden, ohne auf das Endgerät oder den Nutzer angewiesen zu sein. Zusätzlich erlangt er Zugriff auf alle Attribute, die unter Umständen sensible Informationen, wie zum Beispiel die Kreditkartennummer enthalten können. Zu guter Letzt kann er die Token veröffentlichen und somit die Unverfolgbarkeit des Benutzers, die U-Prove normalerweise bietet, aufheben.

4.3. Speicherung im Secure Element

Das Secure Element eignet sich durch seine Sicherheitseigenschaften zur sicheren Speicherung der U-Prove Token. In einem ersten Ansatz wird die komplette Speicherung und

Verarbeitung aller Daten im SE betrachtet. Die Android Applikation ist nur für die Benutzerinteraktion und zum Weiterleiten der APDUs vom Server zum Secure Element zuständig. Für die Kommunikation mit dem JavaCard-Applet (JC-Applet) wird Seek for Android benutzt. Die Lösung ist schematisch in Abbildung 4.3 dargestellt. Zur Freischaltung des Secure Elements gibt der Benutzer die SE-PIN auf dem Smartphone ein. Das anschließende Proving-Protokoll wird genauso wie das Issuing-Protokoll im JC-Applet ausgeführt.

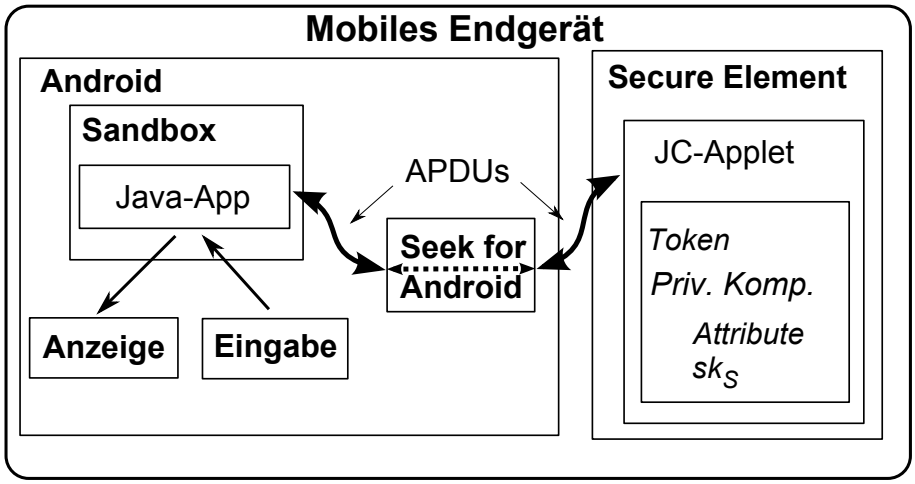

Abbildung 4.3.: Speicherung im SE

Vorteile Gegenüber der vorherigen Lösung besitzt die Speicherung im SE mehrere Sicherheitsvorteile. Als Folge der Annahmen A_3 bis A_5 ist ein unberechtigter Zugriff auf die Daten im SE faktisch nicht möglich. Des Weiteren bietet die PIN einen Schutz gegen unberechtigtes Auslesen solange es dem Angreifer nicht gelingt, eine PIN-Eingabe mitzulesen. Sofern es sich um ein SE im microSD-Format handelt, kann es der Benutzer entfernen und damit unberechtigten Zugriff komplett verhindern. Zusätzlich ist keine Verschlüsselung der Daten notwendig, da das Applet im Secure Element durch die in Abschnitt 2.2.2 genannten Techniken genügend Sicherheit bietet.

Probleme Die beschränkten Ressourcen des Secure Elements stellen eine Herausforderung dar. Kritisch sind insbesondere die geringe Rechengeschwindigkeit der CPU, die geringe Übertragungsrate der Schnittstellen und der geringe Speicherplatz. In2Pay von DeviceFidelity[2] verfügt zum Beispiel als ein typisches Secure Element im microSD-Format über 72KB EEPROM an nichtflüchtigem Speicher und besitzt eine maximale Übertragungsrate von 223.1 kbit/s über die kontaktbehaftete Schnittstelle. Dies steht im Kontrast zu den Anforderungen, die durch die U-Prove Protokolle gestellt werden. Für ein Token kann eine Größe von ca. 1 KB angenommen werden (siehe Abschnitt 2.3.3). 50 Token benötigen somit 50 KB oder auch 60% des Speicherplatzes des In2Pay SE. Nach einer offiziellen

[2] `http://www.devifi.com/in2pay_microsd.html`

Empfehlung sollte ein Applet den Speicherplatz, welchen es während der Laufzeit benötigt, schon bei der Installation reservieren[3]. Dies bedeutet, dass nach der Installation kaum Speicher für andere Applets verfügbar wäre, auch wenn zu dem Zeitpunkt keine Token in dem SE gespeichert sind. Zu guter Letzt lassen die vielen modularen Multiplikationen und Exponentationen der U-Prove Protokolle eine sehr hohe Laufzeit vermuten, da diese mit JavaCard nicht ohne Weiteres auf dem Kryptoprozessor ausgeführt werden können[4].

Aus den genannten Gründen eignet sich das Secure Element nicht zur Speicherung der kompletten Token. Wahrscheinlicher ist ein hybrider Ansatz der im Folgenden erklärt wird.

4.4. Speicherung im Secure Element und Dateisystem

In U-Prove wird auch eine Variante spezifiziert, in der ein Sicherheitsmodul, wie zum Beispiel ein SE, zusätzlich verwendet wird.

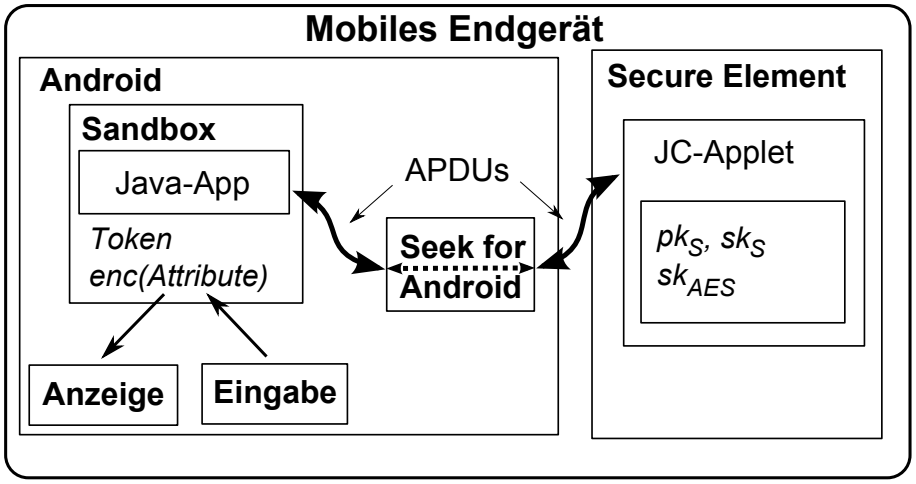

Abbildung 4.4.: Verteilte Speicherung im Secure Element und im Dateisystem

Wie im vorherigen Kapitel baut das Verfahren auf der Java-App und dem JC-Applet (JCA) auf. Im Gegensatz dazu wird im JC-Applet nur ein Teil des U-Prove Protokolls umgesetzt. Das dafür nötige asymmetrische Schlüsselpaar (pk_S, sk_S) wird im Secure Element gespeichert. Dabei handelt es sich um den öffentlichen und privaten Schlüssel des Secure Elements, das kryptografisch mit dem Token verbunden ist. Der Token kann somit nur in Zusammenarbeit mit dem Secure Element präsentiert werden[5]. Des Weiteren enthält die JCA einen symmetrischen Schlüssel sk_{AES}, mit dem die Attribute verschlüsselt

[3] `http://www.oracle.com/technetwork/java/javacard/intro-139322.html#constr`

[4] In [37] wird eine Lösung beschrieben, bei der das gesamte UProve-Protokoll auf der Chipkarte abläuft. Die Lösung basiert hingegen auf dem MULTOS Betriebssystem, welches einen freieren Zugriff auf den Kryptoprozessor erlaubt als Java Card

[5] Siehe dazu [39] S. 18. Dort stellt g_d den öffentlichen und x_d den privaten Schlüssel dar.

werden. Eine Verschlüsselung der Token ist hingegen weniger sinnvoll, da die Token auf dem mobilen Endgerät erzeugt werden und somit zuerst einmal unverschlüsselt auf diesem vorliegen. Eine Verschlüsselung kann somit erst nachträglich durchgeführt werden. Der Zugriff auf das JCA wird durch eine PIN mit Fehlbedienungszähler geschützt, die auf dem Endgerät eingegeben wird. Brute-Force-Angriffe auf die PIN sind somit nicht möglich. Ein Keylogger kann die eingegebene PIN hingegen trotzdem noch abgegreifen.

Vorteile Die Lösung vereinigt die Geschwindigkeitsvorteile der Speicherung im Dateisystem mit den Sicherheitsvorteilen der Speicherung im Secure Element. Bei der Ausstellung und Präsentation der Token findet ein Großteil der Berechnungen auf dem mobilen Endgerät statt. Das Secure Element muss bei der Präsentation nur noch einen Proof of Knowledge durchführen ([39] S. 18.). Auch wird wesentlich weniger Speicherplatz im Secure Element benötigt. Gleichzeitig schützt das Secure Element vor der unerlaubten Vervielfältigung eines Teils der privaten Komponenten. Kopiert ein Angreifer die im Gerät zugänglichen Token und privaten Komponenten, kann er sie trotzdem nicht zur Authentisierung einsetzen, da dazu das Secure Element benötigt wird. Aus den Gründen wird diese Variante für eine Umsetzung favorisiert.

Probleme Die Probleme betreffen vor allem die Sicherheit. Generell hat ein Secure Element Ähnlichkeiten mit einer Chipkarte, die dauerhaft in einem Kartenleser ohne Statusanzeige steckt. Ein Angreifer, der in Besitz der PIN gelangt ist und über eine Verbindung zum mobilen Endgerät verfügt, kann somit die Token nach Belieben zur Authentisierung verwenden.

4.4.1. Sicherheitsbetrachtung

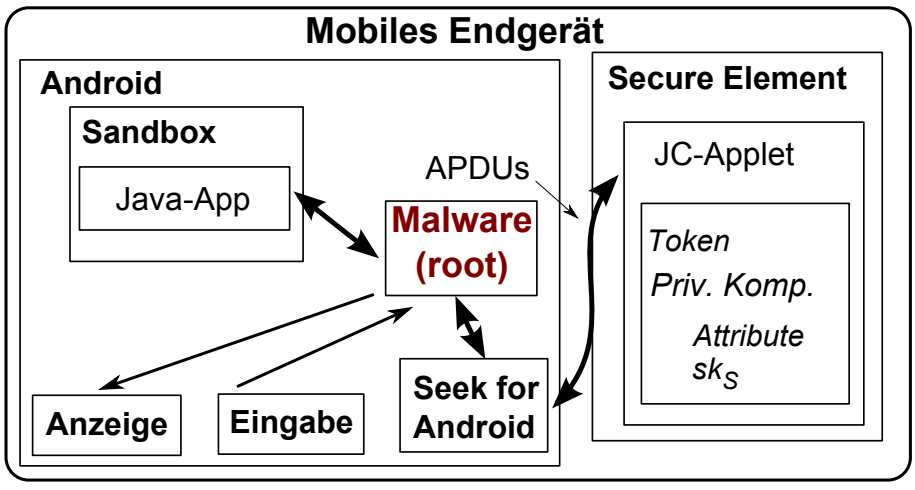

Abbildung 4.5.: Angriff SE

Bei allen folgenden Betrachtungen wird vorausgesetzt, dass der Angreifer Malware mit root-Rechten auf dem Endgerät platziert hat. Diese kann Ein- und Ausgaben und die Kommunikation zwischen Android Applikation und JC-Applet mitlesen sowie verändern und eigene Befehle an das JC-Applet senden.

Sniffing der SE-PIN (** / Verhinderbar)

Beschreibung Der Benutzer authentisiert sich gegenüber dem SE mit seiner PIN, die er auf dem mobilen Endgerät eingibt. Der Angreifer hat mit seiner Schadsoftware nun zwei Möglichkeiten. Er kann die PIN entweder direkt von der Tastatur oder während der Übertragung zum SE abgreifen.

Einstufung Die *Vertraulichkeit* der PIN ist gefährdet. Möchte der Angreifer die mobile eID-Funktion des Secure Elements missbrauchen, benötigt er neben der PIN noch eine Verbindung zum SE des Opfers. Die Gefährdung wird deswegen als **mittel** eingestuft.

Mögliche Gegenmaßnahmen Das Abgreifen der PIN von der Tastatur lässt sich nur dadurch verhindern, dass die Eingabe in einer sicheren Umgebung erfolgt. Dafür existieren zwei Möglichkeiten:

- Das mobile Endgerät verfügt über eine vertrauenswürdige Ausführungsumgebung (Trusted Execution Environment), in der nur privilegierte Programme laufen. Solch ein privilegiertes Programm nimmt die PIN-Eingabe entgegen und leitet sie an das Secure Element weiter.
- Die PIN wird nicht auf dem Gerät sondern auf einem Kartenleser eingegeben, der per NFC mit dem Secure Element kommuniziert.

Beide Gegenmaßnahmen sind auch gegen das Abgreifen während der Übertragung sinnvoll. Zusätzlich ist es auch sinnvoll, die PIN nicht direkt zum SE zu übertragen, sondern nur als Teil eines Challenge-Response-Verfahrens, wie zum Beispiel bei PACE, zu benutzen.

Unberechtigte Authentifizierung (*** / Verhinderbar)

Beschreibung Der Angreifer hat Kenntnis von der SE-PIN erlangt und möchte das Gerät zur eigenen Authentisierung nutzen. Es existieren mehrere Möglichkeiten den Angriff auszuführen. Der Angreifer kann eine eigene Applikation schreiben, mit der er die Kommunikation mit der JCA direkt auf ein eigenes Gerät weiterleitet. Sofern nicht eine eigene Android-Applikation verwendet werden soll, kann der Angreifer ebenfalls das bereits installiert Java-Applet verwenden.

Einstufung Der Angriff stellt einen Missbrauch der mobilen eID-Funktion dar und wird deswegen als **schwerwiegend** eingestuft.

Mögliche Gegenmaßnahmen Eine Zugriffsliste, wie sie zum Beispiel in [22] spezifiziert ist, kann den SE-Zugriff auf eine bestimmte Anwendung festlegen. Somit kann der Angreifer nicht mehr direkt auf das SE zugreifen, sondern muss die vorinstallierte Applikation dafür verwenden. Mit einer TEE kann vorgegebenen werden, dass die PIN-Eingabe über eine (virtuelle) Tastatur des Endgerätes erfolgen muss, auf die der Angreifer hingegen kein Zugriff hat.

Sniffing der aufgedeckten Attribute (*** / Verhinderbar)

Beschreibung Malware, die die Kommunikation zwischen der Android-App und dem JCA mitliest, erhält bei der Nutzerauthentifizierung unter anderem die aufgedeckten Attribute.

Einstufung Die Vertraulichkeit der aufgedeckten Attribute ist gefährdet, sodass die Gefährdung als **schwer** eingestuft wird.

Mögliche Gegenmaßnahmen Eine Verschlüsselung der Attribute im Secure Element für den Diensteanbieter ist hilfreich, sofern das Schlüsselmaterial über einen authentischen Kanal übertragen wird.

Kopieren der Attribute (*** / Verhinderbar)

Beschreibung Das Secure Element kann berechtigte Authentifizierungsvorgänge nicht von unberechtigten Anfragen unterscheiden. Sofern der Angreifer es schafft, eine Abfrage aller Attribute an das SE zu stellen, bekommt er zusätzlich zu dem U-Prove Beweis die Attribute vom SE übermittelt.

Einstufung Die Vertraulichkeit aller Attribute ist gefährdet, sodass die Gefährdung als **schwer** eingestuft wird.

Mögliche Gegenmaßnahmen Geringen Schutz bietet eine Zugriffsliste, in der festgelegt wird, welche Android-Anwendung auf welches JavaCard-Applet zugreifen darf. Eine weitere Möglichkeit ist die Signierung der Anfragen der Android-App, wobei der Schlüssel verschlüsselt und durch bestimmte Maßnahmen schwer auffindbar in der Android-Anwendung fest einkodiert ist. Ebenfalls denkbar ist eine Hardwaresperre des SE, wenn es nicht benötigt wird, z. B. durch dessen Herausnahme, falls es sich um eine microSD-Karte handelt. Am sichersten wäre es, nur die Attributhashes zu übertragen, was die Nutzbarkeit jedoch stark einschränkt. Die Verschlüsselung der Attribute für den Diensteanbieter ist hingegen nutzlos gegen diese Art von Angriff, da der Angreifer sich gegenüber dem SE als Diensteanbieter ausgibt.

Kopieren der Token (* / Wird nicht verhindert)** 

Beschreibung Die Token liegen unverschlüsselt im Dateisystem. Ein Angreifer kann mit Malware die Token kopieren, wodurch die Unverfolgbarkeit des Benutzers nicht mehr gegeben ist.

Einstufung Die Vertraulichkeit der Token wird angegriffen, weswegen die Gefährdung als **schwer** eingestuft wird.

Mögliche Gegenmaßnahmen Wie bereits erwähnt, können die Token vom SE mit dem sk_{AES} Schlüssel verschlüsselt werden. Bei der Tokenausstellung und der Benutzung liegen die Token Prinzip bedingt trotzdem kurzzeitig unverschlüsselt im Arbeitsspeicher des Endgeräts vor.

5. Bereitstellung der Identitätstoken

Im vorherigen Kapitel wurde geklärt, wo die Token gespeichert und verarbeitet werden. Dabei hat sich die gleichzeitige Nutzung des Dateisystems und eines SE als beste Alternative herausgestellt. Unklar ist jedoch noch, wie die Identitätstoken in das unpersonalisierte Endgerät gelangen. Das Problem besteht aus folgenden Teilproblemen:

I. Installation der UI-App im Endgerät

   Die User Interface App (UI-App) ist für die Benutzerinteraktion mit dem Endgerät erforderlich. Dessen Installation kann als weniger sicherheitskritisch als die folgenden Punkte angesehen werden und wird deswegen nicht genauer betrachtet. Unter Android empfiehlt es sich, die UI-App auf Google Play[1], dem offiziellen Portal für Android-Anwendungen, anzubieten. Auch wenn das Programm aus einer vertrauenswürdigen Quelle stammt, muss davon ausgegangen werden, dass es durch Schadsoftware auf dem Endgerät beeinflusst und ausgespäht werden kann.

II. Einrichtung des Secure Elements

   Dieser Schritt beinhaltet die Installation einer SSD für den TSM und des Java Card Applets (JC-Applet) im SE, sofern diese nicht bereits vorhanden sind. Der Einrichtungsprozess lässt sich mit Hilfe von GlobalPlatform Methoden lösen und wird in Abschnitt 5.1 beschrieben.

III. Personalisierung des Secure Elements

   Nach der Einrichtung des SE wurde das Applet zwar installiert, jedoch noch keinem Benutzer zugeordnet. Die Zuordnung, also die Personalisierung, kann entweder in einer vertrauenswürdigen Umgebung beim SE-Herausgeber oder beim Benutzer durchgeführt werden. Während der Personalisierungsprozess bei der ersten Variante durch die vertrauenswürdige Umgebung als sicherer angesehen werden kann, ist die zweite für den Benutzer komfortabler. In Abschnitt 5.2 wird hauptsächlich auf die zweite Variante eingegangen.

IV. Issuing der Identitätstoken

   Beim Issuing werden die Informationen im mobilen Endgerät abgespeichert, mit denen sich der Benutzer später authentisiert. Dabei generieren unter anderem der Token Issuer zusammen mit dem mobilen Endgerät die Identitätstoken. In dessen

[1] `https://play.google.com/store`

Verlauf erzeugt das mobile Endgerät zusammen mit dem Secure Element die benötigten Schlüsselpaare. Der Issuingprozess kann entweder direkt nach der Personalisierung durchgeführt werden oder wenn die bereits in dem Endgerät bestehenden Token aufgebraucht sind bzw. deren Gültigkeit abgelaufen ist. Die bereits personalisierten Informationen können als Anker dienen, um sicherzustellen, dass das Issuing mit dem korrekten Gerät durchgeführt wird. Das Issuing wird in Abschnitt 5.3 betrachtet.

5.1. Einrichtung des Secure Elements

Bevor das Issuing der Identitätstoken durchgeführt werden kann, muss das Secure Element eingerichtet werden, wobei die Einrichtung in zwei Schritten erfolgt. Im ersten Schritt wird die TSM-SSD installiert, die als Repräsentation des TSM je nach erhaltenen Privilegien bestimmte administrative Aufgabe im SE ausführen kann. Danach erfolgt die Installation des unpersonalisierten Java Card Applets (JC-Applet). Die Personalisierung, also die Zuordnung zu einem bestimmten Benutzer, soll erst später erfolgen. Bei der Installation des JC-Applet muss vor allem sichergestellt werden, dass es vertrauenswürdig ist und während der Übertragung nicht verändert wurde.

In Abbildung 5.1 wird das ausgelieferte SE aus GlobalPlatform-Sicht dargestellt. In ihm befindet sich nur die ISD mit den gesetzten Kartenschlüsseln *S-ENC*, *S-MAC* und *DEK*.

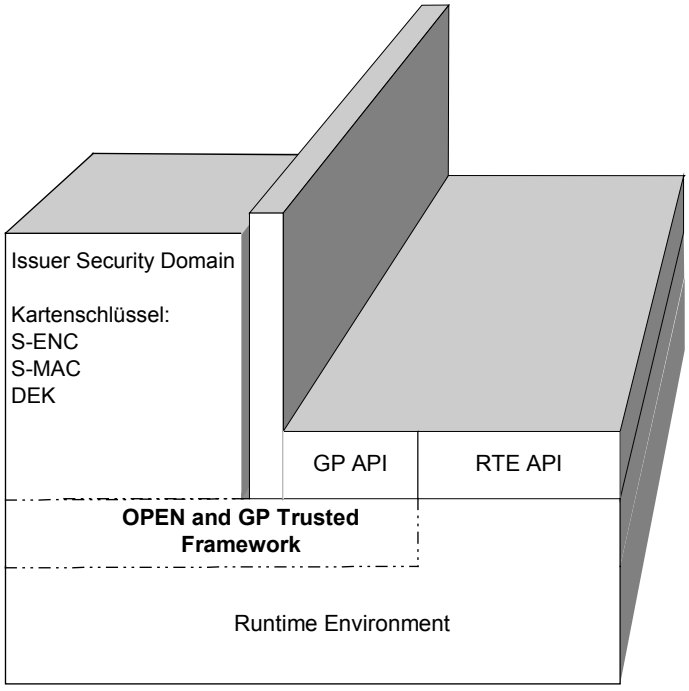

Abbildung 5.1.: SE im Auslieferungszustand aus der Sicht von GlobalPlatform - nach [18]

Die Einrichtung des SE wird in die Installation einer SSD für den TSM und des JC-Applets unterteilt. Dafür müssen folgende **Voraussetzungen** gegeben sein:

- Das Secure Element ist GlobalPlatform 2.1.1 kompatibel.
- Es befindet sich im Status *SECURED*. Die Schlüssel der Issuer Security Domain wurden also vom Issuer bereits neu gesetzt, weswegen nur er das SE administrieren kann.
- Das SE muss mit der bereits auf dem Endgerät installierten UI-App kommunizieren können, wobei der SE-Typ nicht von Bedeutung ist.
- Sollte der TSM bereits über eine eigene SSD im SE verfügen, kann der Abschnitt 5.1.1 übersprungen werden.

5.1.1. Einrichtung der TSM-SSD

Existiert in dem SE noch keine Supplementary Security Domain (SSD) des zuständigen TSM, muss diese instanziiert werden, wofür die Zusammenarbeit mit dem ISD-Inhaber erforderlich ist. Die SSD repräsentiert den TSM auf der Karte und kann verschiedene administrative Aufgaben auf dem SE durchführen, sofern ihr die entsprechenden Privilegien (z. B. DAP-Verification oder Delegated Management) vom ISD-Inhaber gegeben wurden. Das Sequenzdiagramm in Abbildung 5.2 stellt den Einrichtungsvorgang dar, wobei davon ausgegangen wird, dass die UI-App bereits installiert wurde und mit dem SE kommunizieren kann.

(a) Die UI-App versucht die SSD des TSM zu selektieren.

(b) Da keine SSD vorhanden ist, wird eine Installationsanforderung an den TSM geschickt, welche Informationen zur Ermittlung des ISD-Inhabers (ISD-I), z. B. die IIN oder CIN, enthält.

(c) Der TSM überprüft, ob er für das SE zuständig ist und ermittelt gegebenenfalls den zugehörigen ISD-Inhaber.

(d) Der ISD-Inhaber selektiert die ISD bevor sich ISD-Inhaber und ISD gegenseitig per Challenge-Response-Verfahren mit dem *S-ENC* Schlüssel authentifizieren. Dabei werden der TSM und die UI-App als Zwischenstationen genutzt. Als Ergebnis entsteht ein verschlüsselter und mit einem MAC abgesicherter Secure-Messaging-Kanal zwischen den beiden Instanzen.

(e) Über diesen Kanal sendet der ISD-Inhaber ein *Install [for Install] SSD* Kommando an die ISD, welches als Parameter entweder DAP-Verification oder DM-Privilegien enthält. Diese instanziiert im SE die neue Security Domain *SSD* mit den übermittelten Privilegien.

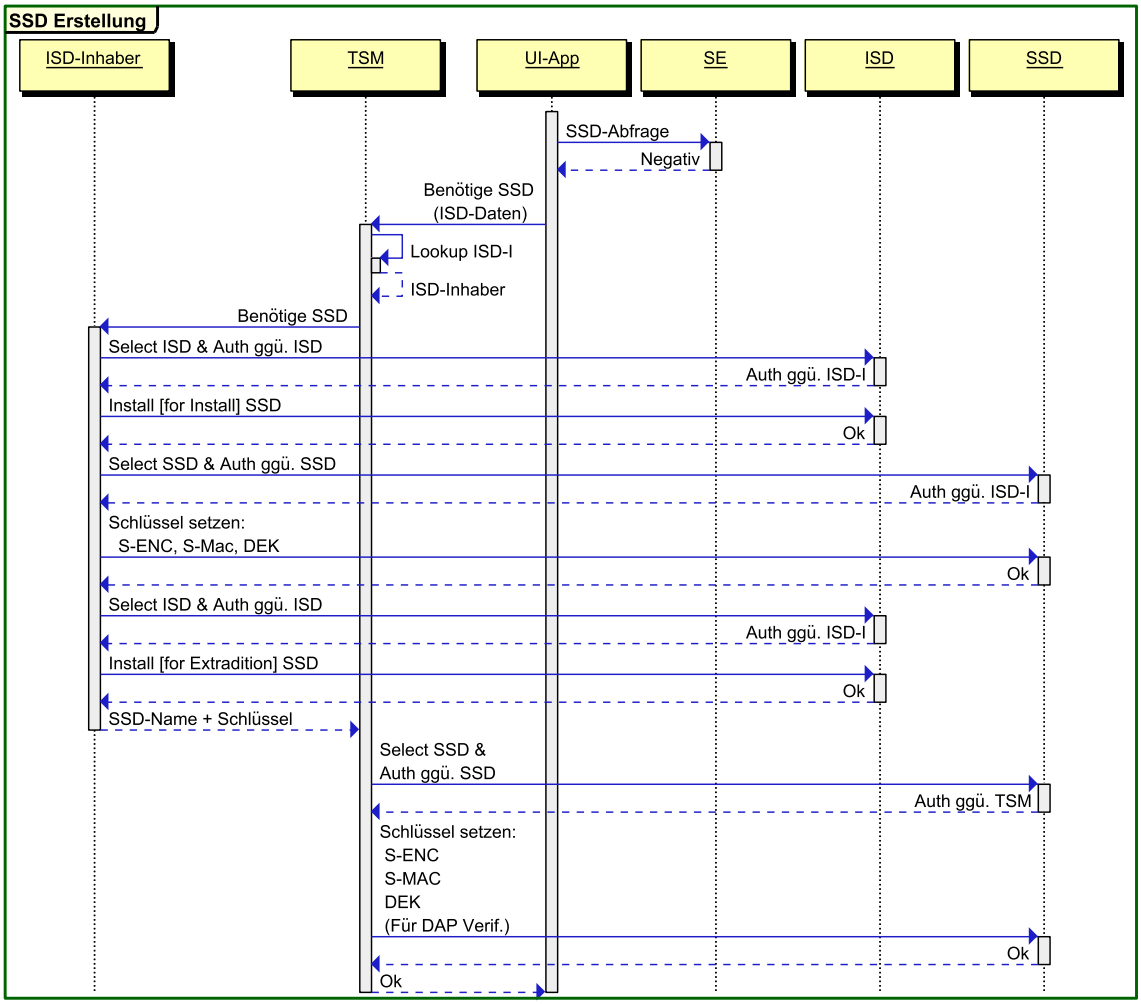

Abbildung 5.2.: SSD Erstellung

(f) Der ISD-Inhaber selektiert die SSD, woraufhin wieder eine gegenseitige Authentifizierung stattfindet. Anschließend setzt er die Schlüssel S-ENC, S-MAC und DEK der SSD neu, wobei diese zufällig gewählt sein müssen. Unterstützt das SE das Ammendment A der GlobalPlatform Card Specification[17], sollte der ISD-Inhaber zusätzlich den öffentlichen Schlüssel des TSM einspielen. In beiden Fällen müssen die neu gesetzten Schlüssel über einen sicheren Kanal an den TSM geleitet werden.

(g) Anschließend selektiert der ISD-Inhaber wieder die ISD und sendet ein *Install [for Extradition] SSD* Kommando an sie, wodurch die SSD zu einer eigenständigen Security Domain wird.

(h) Mit den erhaltenen Schlüsseln baut der TSM einen Secure Messaging geschützten Kanal zur SSD auf, über den er die Schlüssel durch eigene ersetzen kann. Besitzt die

SSD das DAP-Verification Privileg, wird zusätzlich der öffentliche DAP-Verification Schlüssel eingespielt.

Sicherheitsbetrachtung

Ein Großteil des Ablaufes basiert auf der GlobalPlatform Kartenspezifikation V.2.1.1. [18]. Diese bietet zwar eine textuelle Beschreibung für die SSD Einrichtung, jedoch wird kein konkreter Ablauf mit einzelnen Befehlen spezifiziert, weswegen ebenfalls keine Sicherheitsheitsanalyse dazu existiert. Besonders kritisch ist dabei die Übergabe der temporären SSD-Schlüssel vom ISD-Inhaber zum TSM, die allerdings nach Annahme A_8 geschützt erfolgt. Weiterhin stellen die Select-Befehle und der Get-Data-Befehl eine Schwachstelle dar, da diese nicht per Secure Messaging geschützt werden.

Umleitung in falsches SE (* / Wird nicht verhindert)

Beschreibung: Der Angreifer leitet die Verbindung zwischen UI-App und SE des Opfers in ein eigenes Secure Element um. Folglich wird die Security-Domain-Installation mit dem SE des Angreifers ausgeführt. Befindet sich dieses im selben Status, wie das SE des Opfers (also ohne installierte SSD), kann nur ein technisch versierter Benutzer diesen Angriff erkennen, indem er die Netzwerkkommunikation oder die Kommunikation mit dem eigenen Secure Element überwacht. Durch GlobalPlatform-Mechanismen wird nur sichergestellt, dass die Verbindung in einem Secure Element terminiert.

Einstufung: Der Angriff dient als Vorbereitung für die Umleitung der Applet-Installation, die für den Identitätsdiebstahl verwendet wird und wird somit als **leicht** eingestuft. Der Angriff zielt auf die Authentizität des Secure Elements ab.

Mögliche Gegenmaßnahmen: Bisher existiert noch keine Technologie, die vor dieser Art Angriff vollständig schützt. Folgende Vorschläge wären denkbar, wobei ersterer nur der Information des Benutzers dient, dass er gerade angegriffen wird.

- Eine TEE verbindet das Secure Element mit einer vertrauenswürdigen Anzeige. Über diese erhält der Benutzer Rückmeldung über den Fortschritt des Installationsvorgangs. Weicht die Anzeige von der in der UI-App ab, weiß er, dass etwas nicht stimmt.

- Erhält der TSM Informationen über das SE sowie über das mobile Endgerät des Benutzers, sollte er einen Plausibilitätscheck durchführen, ob die Kombination von SE und Endgerät realistisch ist. Beispielsweise verfügen manche Smartphones über keinen microSD-Anschluss, wodurch das Einstecken eines SE im microSD-Format technisch nicht möglich ist. Durch die Wahl eines geeigneten Secure Elements kann der Angreifer diese Gegenmaßnahme jedoch umgehen.

Umleitung zu eigenem Server (* / Wird verhindert)

Beschreibung Der Angreifer leitet die Anfrage der UI-App zu einem eigenem Server um und gibt sich als TSM aus. Sofern er über eine Verbindung zum ISD-Inhaber verfügt, kann er sich über ihn eine eigene SSD generieren lassen.

Einstufung Der Angriff dient als Vorbereitung für die Umleitung der Applet-Installation, die wiederum den Identitätsdiebstahl vorbereitet und wird somit als **leicht** eingestuft. Der Angriff zielt auf die Authentizität des TSM ab.

Mögliche Gegenmaßnahmen Nach der Annahme A_8 besitzen TSM und ISD-Inhaber ein vertragliches Verhältnis. Erhält der ISD-Inhaber von einer ihm unbekannten Quelle eine Anfrage, muss er diese ablehnen.

Replay Angriff (* / Wird verhindert)

Beschreibung Der Angreifer zeichnet die gesamte Kommunikation zwischen der UI-App und dem SE auf und versucht, eine eigene SSD im SE zu installieren, indem er die aufgezeichneten APDUs entsprechend abändert und erneut an das SE sendet.

Einstufung Der Angriff dient als Vorbereitung für die Umleitung der Applet-Installation, die wiederum den Identitätsdiebstahl vorbereitet und wird somit als **leicht** eingestuft. Der Angriff zielt auf die Authentizität des ISD-Inhabers und die Integrität der Installationsbefehle ab.

Mögliche Gegenmaßnahmen Der Angriff wird durch die External Authentication verhindert, da dabei das SE und das Terminal eine zufällige Challenge erzeugen. Spielt der Angreifer die Befehle wieder ein, scheitert er somit an der Authentifizierung, da er nicht auf die Zufallszahl des SE reagieren kann.

Mitlesen eines Select-Befehls (* / Wird nicht verhindert)

Beschreibung Select-Befehle werden prinzipbedingt nicht durch Secure Messaging geschützt und somit nicht verschlüsselt. Folglich kann Schadsoftware diese mitlesen und kennt somit die AID der ISD und der SSD.

Einstufung Die Vertraulichkeit der jeweiligen AID ist gefährdet. Mit deren Kenntnis kann der Angreifer einen Brute-Force-Angriff auf die Authentifizierung der Security Domain durchführen, sodass die Gefährdung als **leicht** eingestuft wird.

Mögliche Gegenmaßnahmen Nach GlobalPlatform ist keine Verschlüsselung der Select-Befehle vorgesehen. Um diese zu unterstützen, müsste sich eine den Security Domains vorgeschaltete Instanz im SE (z. B. die OPEN oder das Betriebssystem) um die Authentifizierung und das anschließende Secure Messaging kümmern.

Verändern eines Select-Befehls (/ Wird verhindert)**

Beschreibung Select-Befehle werden prinzipbedingt nicht durch Secure Messaging geschützt und können folglich verändert werden. Konkret kann der Angreifer den Select der SSD, bei dem der ISD-Inhaber die temporären SSD-Schlüssel setzen möchte (Schritt (f) im Ablauf), in einen Select der ISD umändern. Da in dem Moment SSD und ISD dieselben Schlüssel zur Authentifizierung verwenden, authentifiziert sich der ISD-Inhaber somit gegenüber der ISD und ersetzt deren Schlüssel durch die temporären Schlüssel.

Einstufung Die Integrität der Select-Befehle ist gefährdet[2]. Sofern die temporären Schlüssel ein geringeres Sicherheitsniveau als die bestehenden ISD-Schlüssel besitzen, kann der Angriff als Vorbereitung für den Brute-Force-Angriff auf die Authentifizierung gegenüber der ISD angesehen werden und hat deswegen eine **mittlere** Gefährdung[3].

Mögliche Gegenmaßnahmen Nach der Authentifizierung sollte ein Befehl gesendet werden, der garantiert unterschiedlich von ISD und SSD beantwortet wird. Da die SSD im Gegensatz zur ISD noch nicht eingerichtet ist, bietet sich das *GET STATUS* Kommando, das den aktuellen Stand im Lebenszyklus der Security Domain abfragt, an.

Brute-Force-Angriff auf die Authentifizierung (* / Wird verhindert)**

Beschreibung Der Angreifer versucht, sich gegenüber einer Security Domain ohne Kenntnis der Schlüssel zu authentifizieren. Der Angriff verspricht am meisten Erfolg nachdem die temporären Schlüssel der SSD (oder nach dem Angriff *Verändern eines Select-Befehls* der ISD) gesetzt wurden.

Einstufung Der Angriff zielt auf die Authentizität des ISD-Inhabers oder des TSM ab. Gelingt er bei der ISD, erhält der Angreifer die volle Kontrolle über das SE, sodass die Gefährdung als **schwer** eingestuft wird.

Mögliche Gegenmaßnahmen Grundsätzlich gilt nach der Annahme A_4, dass die Authentifizierung nicht umgangen werden kann.

- Alle Security Domain Schlüssel müssen zufällig gewählt sein.
- Bereits jetzt besitzen manche Secure Elements einen Fehlbedienungszähler, der die entsprechende Security Domain nach einer bestimmten Zahl an fehlerhaften Authentifizierungen sperrt.

[2] Zusätzlich ist die Verfügbarkeit des SE gefährdet, da sich der ISD-Inhaber hinterher nicht mehr gegenüber der ISD authentifizieren kann. Der Grundwert wird jedoch wegen den in Abschnitt 3.2 genannten Gründen nicht betrachtet

[3] Da nach Annahme A_7 der TSM vertrauenswürdig ist, nutzt er die Lücke nicht aus. Andernfalls müsste der Angriff als **schwerwiegend** eingestuft werden.

- Eine Zugriffsbeschränkung des SE nach [22] kann unberechtigte Zugriffe auf das SE ebenfalls verbieten. Wird der Angriff mit *Umleitung zu eigenem Server* kombiniert, kann die Beschränkung allerdings umgangen werden.

5.1.2. Installation des JC-Applets

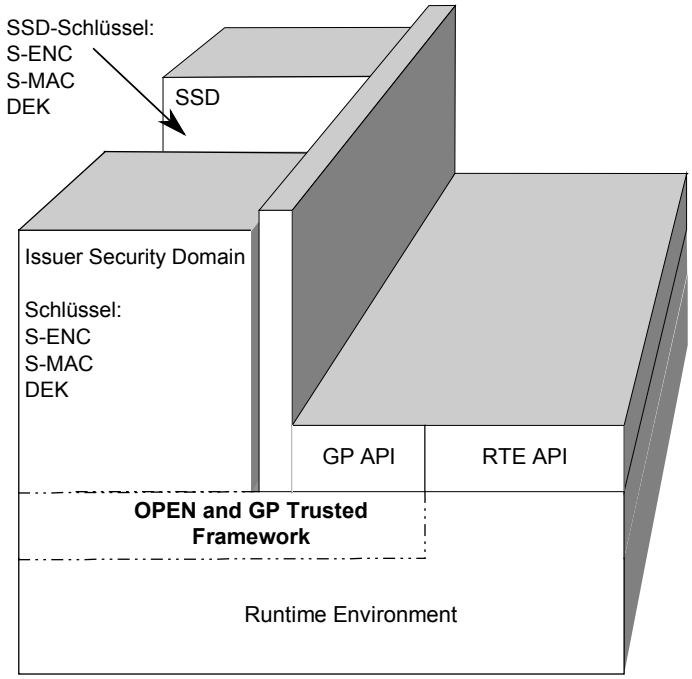

Abbildung 5.3.: SE mit eingerichteter SSD - nach [18]

Die Applet-Installation kann auf zwei Wegen erfolgen, je nachdem welche Rechte die SSD des TSM besitzt. Da in dem JCOP-Betriebssystem eine SSD maximal nur DAP-Verifikation Rechte besitzen darf, wird dieser Fall hier betrachtet[4]. Die Besonderheit ist dabei, dass der ISD-Inhaber für die Installation zuständig ist. Die TSM-SSD überprüft nur nach dem Hochladen des JC-Applets in das SE die Integrität des Applets anhand der mitgelieferten Signatur. Ist diese korrekt, wird das Applet installiert.

(a) Die UI-App versucht das JC-Applet zu selektieren.

(b) Da das Applet nicht vorhanden ist, wird eine Installationsanforderung an den TSM geschickt, die Informationen zur Ermittlung des ISD-Inhabers (ISD-I), z. B. die IIN oder CIN, enthält.

(c) Der TSM überprüft anhand der Informationen, ob er für das SE zuständig ist und ermittelt den zugehörigen ISD-Inhaber.

[4]Der Installationsablauf mit Delegated Management ist im Anhang (Abbildung A.1) zu sehen.

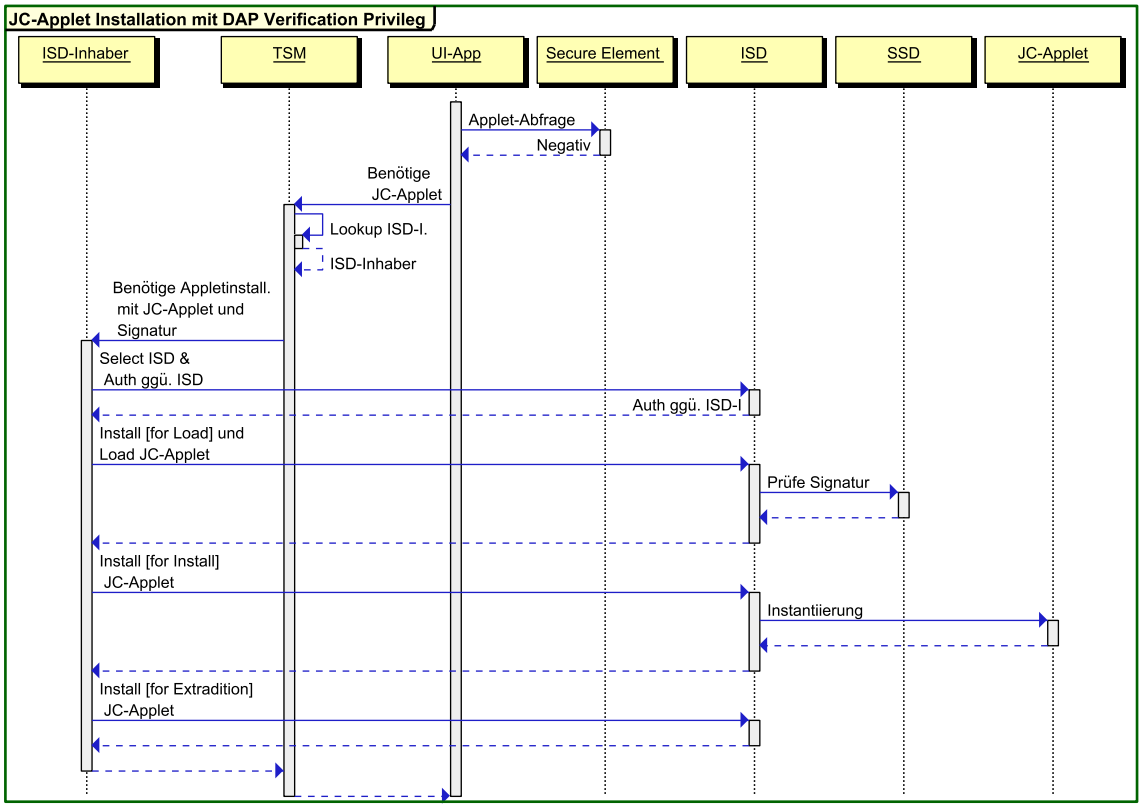

Abbildung 5.4.: JC-Applet Installation mit DAP-Verifikation

(d) Neben der Installationsanforderung übermittelt der TSM dem ISD-Inhaber das von ihm signierte Applet.

(e) ISD-Inhaber und ISD authentifizieren sich gegenseitig mit dem *S-ENC* Schlüssel, wobei der TSM und die UI-App als Zwischenstationen genutzt werden. Als Ergebnis entsteht ein verschlüsselter und mit einem MAC abgesicherter Secure-Messaging-Kanal zwischen den beiden Instanzen.

(f) Über diesen Kanal sendet der ISD-Inhaber ein *Install [for Load]* und ein *Load JC-Applet* Kommando an die ISD, mit denen das JC-Applet in das Secure Element gespielt wird.

(g) Im SE leitet die OPEN das Applet an die SSD des TSMs weiter, welche mit dem DAP-Verification Schlüssel die Applet-Signatur überprüft. Ist diese korrekt, wird das Applet im SE gespeichert, wobei die ISD als verantwortliche Security Domain in der Card Registry registriert wird.

(h) Anschließend sendet der ISD-Inhaber ein *Install [for Install]* und ein *Install [for Extradition]* Kommando, mit denen das Applet installiert bzw. instanziiert und die SSD als verantwortliche Security Domain registriert wird. In diesem Schritt können

auch Installationsparameter übertragen werden, die an den Konstruktor des Applets weitergeleitet werden. So kann beispielsweise eine PIN gesetzt werden, mit der sich der Benutzer später gegenüber dem Applet authentisiert.

Nachdem die Punkte abgearbeitet wurden, ist der Installationsvorgang für den ISD-Inhaber sowie für den TSM abgeschlossen. Da das Applet noch keine personenbezogenen Daten enthält, muss es anschließend für den Benutzer personalisiert werden.

Sicherheitsbetrachtung

Umleitung in falsches SE (** / Wird nicht verhindert)

Beschreibung: Dieser Angriff ist sehr ähnlich zu dem Umleitungsangriff auf die Einrichtung der TSM-SSD. Auch hier leitet der Angreifer die Verbindung zwischen UI-App und SE in ein eigenes Secure Element um. Folglich wird die Appletinstallation mit dem SE des Angreifers ausgeführt.

Einstufung: Der Angriff dient als Vorbereitung für den Identitätsdiebstahl und wird somit als **mittel** eingestuft. Für den endgültigen Angriff muss noch die Personalisierung umgeleitet werden. Der Angriff zielt auf die Authentizität des Secure Elements ab.

Mögliche Gegenmaßnahmen: Die bereits beim Angriff *Umleitung in falsches SE* in Abschnitt 5.1.1 vorgestellten möglichen Gegenmaßnahmen treffen auch hier zu. Somit gilt hier ebenfalls, dass noch keine effektive Gegenmaßnahme existiert.

Umleitung der Applet-Installation (*** / Wird verhindert)

Beschreibung Der Angreifer leitet die Anfrage der UI-App zu einem eigenem Server um und gibt sich als TSM aus. Sofern er über eine Verbindung zum ISD-Inhaber verfügt, kann er über ihn ein eigenes Applet im SE installieren.

Einstufung Besitzt der Angreifer ein eigenes Applet im SE, kann er die privaten Daten und Token des Benutzers auf ein eigenes Gerät kopieren und somit seine Identität stehlen, sodass die Gefährdung als **schwer** eingestuft wird. Der Angriff zielt auf die Authentizität des TSM ab.

Mögliche Gegenmaßnahmen Nach der Annahme A_8 besitzen TSM und ISD-Inhaber ein vertragliches Verhältnis. Erhält der ISD-Inhaber von einer ihm unbekannten Quelle eine Anfrage, muss er diese ablehnen. Des Weiteren stellt die DAP Verification der TSM-SSD sicher, dass nur Applets mit gültiger Signatur in dem SE installiert werden. Die Maßnahme ist jedoch nicht wirksam, wenn der Angreifer bereits eine eigene SSD mit DAP-Verification-Privileg besitzt.

Mitlesen des Load-Befehls (* / Wird verhindert)**

Beschreibung: Der Angreifer liest das übertragene Applet mit. Auch wenn es sich in einer komprimierten und kompilierten Form befindet, kann er durch einen Java Disassembler eventuell an geheime Informationen gelangen.

Einstufung: Normalerweise sollte eine Anwendung nicht aufgrund der Geheimhaltung des Codes sicher sein, nichtsdestotrotz ist die Vertraulichkeit der Anwendung gefährdet. Die Gefährdung ist von Applet zu Applet unterschiedlich und schwankt je nach Programmqualität zwischen **leicht** und **schwer**.

Mögliche Gegenmaßnahmen: Das Load-Kommando sollte per Secure Messaging geschützt sein. Befindet sich das SE in einem initialisierten Zustand, wird dies nach eigenen Erkenntnissen von JCOP erzwungen.

Verändern eines Load-Befehls (* / Wird verhindert)**

Beschreibung Der Angreifer verändert die Load-Befehle, in denen das JC-Applet übertragen wird und schleust darüber ein eigenes Applet in das SE ein.

Einstufung Die Folge ist die selbe wie bei der *Umleitung der Applet-Installation*. Besitzt der Angreifer ein eigenes Applet im SE, kann er die privaten Daten und Token des Benutzers auf ein eigenes Gerät kopieren und somit seine Identität stehlen, sodass die Gefährdung als **schwer** eingestuft wird. Der Angriff zielt auf die Integrität des Applets ab.

Mögliche Gegenmaßnahmen Es existieren zwei Möglichkeiten, um die Integrität des Applets während der Übertragung sicherzustellen.

- Bei der Appletübertragung muss die vom TSM ausgestellte Signatur in Form eines DAP mitgesendet werden. Mit der Überprüfung des DAP kann die TSM-SSD die Integrität des Applets überprüfen.
- Das Load-Kommando muss mit einem MAC gegen Veränderungen geschützt werden.

5.2. Personalisierung des Secure Elements im Feld

Nachdem das Applet erfolgreich in das SE geladen und dort installiert wurde, muss es noch personalisiert werden, was entweder in einer vertrauenswürdigen Umgebung beim SE-Herausgeber oder beim Benutzer („Im-Feld-Personalisierung“) durchgeführt werden kann. Durch die vertrauenswürdige Umgebung kann der Personalisierungsprozess bei der ersten Variante generell als sicherer als bei der zweiten angesehen werden. Die Wahrscheinlichkeit, dass ein Angreifer die zu personalisierenden Daten abgreift oder verändert oder sogar eigene Daten in das SE bringt, ist hier geringer. Die zweite Variante ist hingegen komfortabler für den Anwender, da dieser nicht auf die Lieferung eines vorpersonalisierten Secure Elements

warten oder vielleicht sogar sein Endgerät mit integriertem SE zur Personalisierung zum Herausgeber schicken muss. Im Folgenden wird die zweite Personalisierungsart betrachtet. Dabei handelt es sich um den komplexesten der zu Beginn von Kapitel 5 beschriebenen Punkte, da die sichere Personalisierung im Feld eine Art „Henne-Ei-Problem“ darstellt:
Erst durch die Personalisierung kann ein Secure Element einem Benutzer zugeordnet werden. Diese Zuordnung muss jedoch bereits während der Personalisierung überprüft werden, um sicherstellen zu können, dass die Verbindung mit dem korrekten SE aufgebaut wurde. Wegen diesem Problem ist die Personalisierung anfällig für bestimmte Angriffe, wie zum Beispiel der Umleitung.

5.2.1. Personalisierungsumleitung

In den vorherigen Abschnitten zur *Einrichtung der TSM-SSD* 5.1.1 und *Installation des JC-Applets* 5.1.2 wurde in der jeweiligen Sicherheitsbetrachtung die Umleitung der Daten in ein vom Angreifer kontrolliertes SE betrachtet. Deren Bedrohungen wurden als leicht bzw. mittel eingestuft, wobei keine effektiven Gegenmaßnahmen existieren. Um den Identitätsdiebstahl zu verhindern, muss somit die Personalisierung eine Gegenmaßnahme implementieren. Bei genauerer Betrachtung basiert die Personalisierungsumleitung auf zwei Teilproblemen:

- Ein Angreifer kann die Kommunikation in sein eigenes SE umleiten.
- Die Umleitung kann vor einem Benutzer verschleiert werden.

Die erste Bedrohung kann prinzipbedingt nicht ausgeschlossen werden, wenn die Personalisierung beim Benutzer erfolgt. Ursächlich dafür sind die unsicheren Geräte, wie Desktoprechner oder mobiles Endgerät, die der Benutzer besitzt, um eine Verbindung mit dem SE herzustellen. Befindet sich darauf Schadsoftware, kann sie, wie in Abbildung 5.5 zu sehen ist, die Verbindung zu einem externen vom Angreifer kontrollierten SE umleiten.
Die zweite Bedrohung basiert darauf, dass Secure Elements aktuell weder über eine sichere Eingabe oder Ausgabe noch über eine Aktivitätsanzeige verfügen. Der Benutzer kann somit nur schwer überprüfen, ob die Kommunikation tatsächlich mit seinem SE stattfindet und bei einem Angriff die Personalisierung abbrechen.

Eine mögliche Lösung ist es, das SE direkt[5] per NFC mit einem Standard- oder Komfortleser kommunizieren zu lassen. Deren vertrauenswürdige Anzeige kann dazu genutzt werden, dem Benutzer eine Rückmeldung über den Personalisierungsvorgang zu geben. Die Ein- und Ausgabe eines solchen Lesers kann mit Hilfe spezieller Standards wie Secoder[6] oder EAC genutzt werden. Da für Secoder keine öffentlichen Quellen verfügbar sind, EAC durch das System neuer Personalausweis dafür um so besser dokumentiert ist, fällt die Wahl in der vorliegenden Arbeit auf zweiteres.

[5] Hierbei meint *direkt*, dass die Kommunikation höchstens noch über den NFC-Controller geleitet wird. Das Betriebssystem des mobilen Endgeräts erhält hingegen keine Informationen.

[6] `http://www.die-deutsche-kreditwirtschaft.de/dk/zahlungsverkehr/kartengestuetzter-zahlungsverkehr/secoder.html`

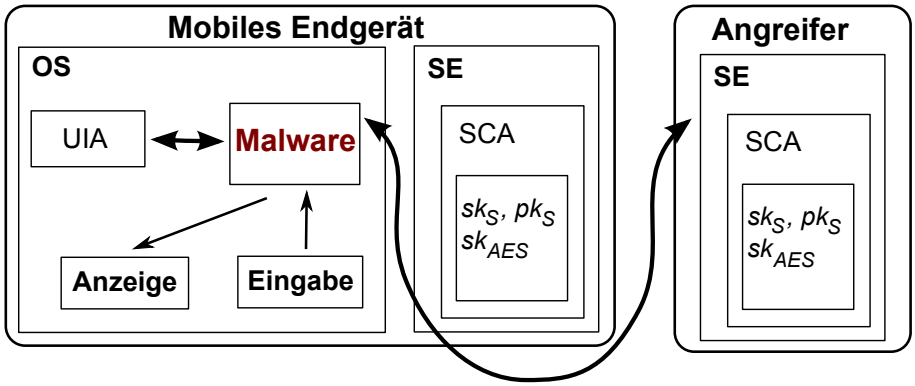

Abbildung 5.5.: Umleitung der Personalisierung

5.2.2. Personalisierung und Verifikation mit EAC

Die Personalisierung wird in zwei Phasen unterteilt. In der ersten Phase findet die eigentliche Personalisierung statt, bei der der meID-Server mit Hilfe der EAC eine sichere Verbindung zum SE aufbaut. Neben dem Identifizierer, den er aus einer authentischen Datenquelle des Benutzers erhalten hat, schreibt der meID-Server außerdem noch eine zufällig gewählte Nonce[7] in das SE. Anschließend findet eine Verifikations- und Freischaltungsphase statt, in der die Personalisierungsdaten neu aus der authentischen Datenquelle ausgelesen und anschließend zusammen mit der Nonce mit den personalisierten Daten des SE verglichen werden. Fängt ein Angreifer die erste Personalisierungsphase ab, merkt der Benutzer dies und führt die anschließende Verifikations- und Freischaltungsphase nicht aus. Das SE des Angreifers wird somit nicht freigeschaltet. Das komplette dazugehörige Sequenzdiagramm befindet sich auf Grund seiner Größe im Anhang (Abbildung A.2).

Personalisierung

Die erste Phase der Personalisierung wird im Sequenzdiagramm 5.6 dargestellt. Der in der Abbildung grün umrandete Bereich *eID nPA 1* stellt dabei den bereits bekannten Auslesevorgang aus dem nPA dar. In *eID SE 1* findet der Personalisierungsvorgang des SE statt, wobei das SE, genau so wie der nPA, erst nach einem EAC-Durchlauf den Zugriff gewährt. SE und nPA befinden sich in unterschiedlichen PKIs, weswegen das SE kein Berechtigungszertifikat für den nPA akzeptiert und umgekehrt.

(a) Der Benutzer fordert mit seinem Computer eine Personalisierung auf der Webseite des meID-Servers an.

(b) Diese antwortet mit einem SAML-Request, der an den eID-Server gerichtet ist, wodurch der in Abschnitt 2.3.1 beschriebene Ablauf angestoßen wird.

[7] Bei einer Nonce handelt es sich um eine einmalig verwendete Zahl.

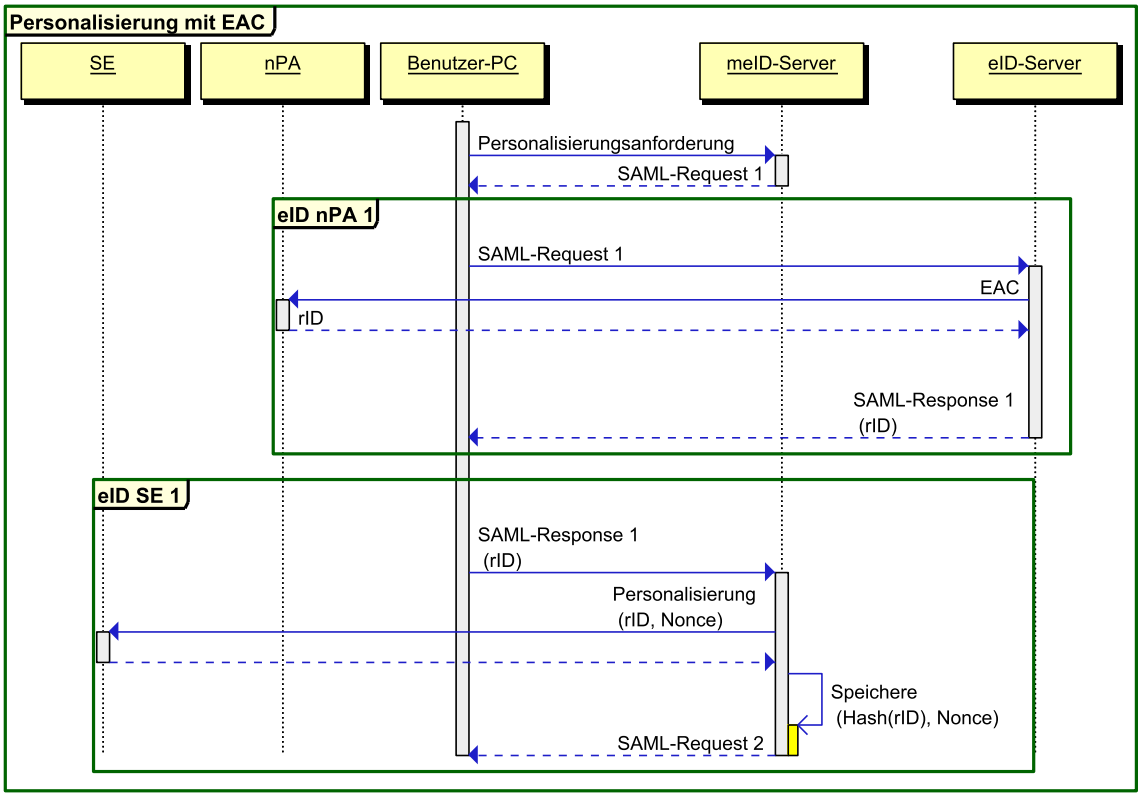

Abbildung 5.6.: Personalisierung mit EAC

(c) An deren Ende erhält der meID-Server den restricted Identifier (rID) und eventuell weitere Attribute[8], die für den Identifier verwendet werden.

Daraufhin wird die eigentliche Personalisierung durchgeführt, die in Abbildung 5.6 in der Box *eID SE 1* zu sehen ist. In Abbildung 5.7 ist dieser Schritt detaillierter dargestellt.

(d) Die meID-Anwendung baut eine TLS-RSA-PSK gesicherte Verbindung zum meID-Server auf, über die der meID-Server Kommandos an das SE senden kann.

(e) Dieser sendet das CV-Zertifikat , welches die Schreibberechtigung für die Personalisierung enthält, an die Anwendung.

(f) Das Zertifikat wird mit einem Standard- oder Komfortleser, der über die kontaktlose Schnittstelle mit dem SE kommuniziert, an das SE weitergeleitet.

(g) Nachdem der Benutzer sich die Zertifikatsberechtigungen auf dem Leser angesehen hat, bestätigt er den Vorgang durch Drücken der OK-Taste auf dem Leser. Daraufhin

[8] Um den Ablauf möglichst einfach zu halten, wird im Folgenden und in den Sequenzdiagrammen nur die rID verwendet.

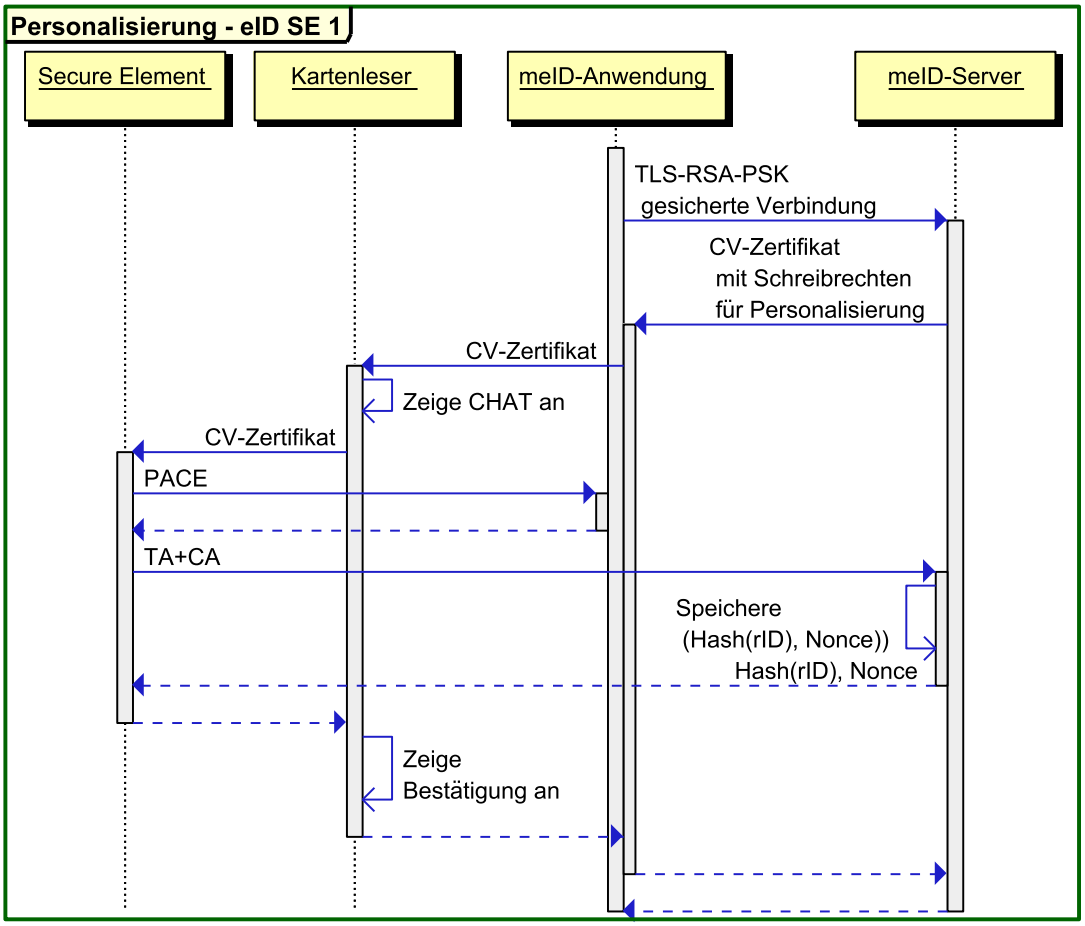

Abbildung 5.7.: Personalisierung mit EAC - eID SE 1

führen das Secure Element und die meID-Anwendung PACE mit einer Standard-CAN durch.

(h) Der meID-Server und das SE authentifizieren sich bei der Terminal- und Chip-Authentication gegenseitig, an deren Ende ein per Secure Messaging gesicherter Kanal zwischen den beiden Akteuren besteht. Der erfolgreiche Durchlauf der EAC wird auf dem Kartenleserdisplay angezeigt.

(i) Der meID-Server hasht die rID und speichert sie zusammen mit einer zufällig gewählten Nonce für die Dauer der Session in einer Datenbank ab. Die Daten müssen während der Verifikationsphase oder nach Eintreten einer Zeitüberschreitung wieder gelöscht werden.

(j) Über den sicheren Kanal findet die Personalisierung statt, bei der die rID zusammen mit einer vom meID-Server zufällig gewählten Nonce an das SE übertragen wird.

Verifikation und Freischaltung

In der Verifikationsphase überprüft der meID-Server, ob das korrekte SE personalisiert wurde. Der Ablauf ist ähnlich dem der Personalisierung, nur dass der meID-Server diesmal die Nonce aus der Datenbank ausliest und zusammen mit der gehashten rID an das SE sendet. Stimmt das Tupel mit dem in dem SE gespeicherten überein, wird das SE freigeschaltet und kann für die mobile Authentifizierung genutzt werden. Der genaue Ablauf ist im Sequenzdiagramm in Abbildung 5.8 zu sehen.

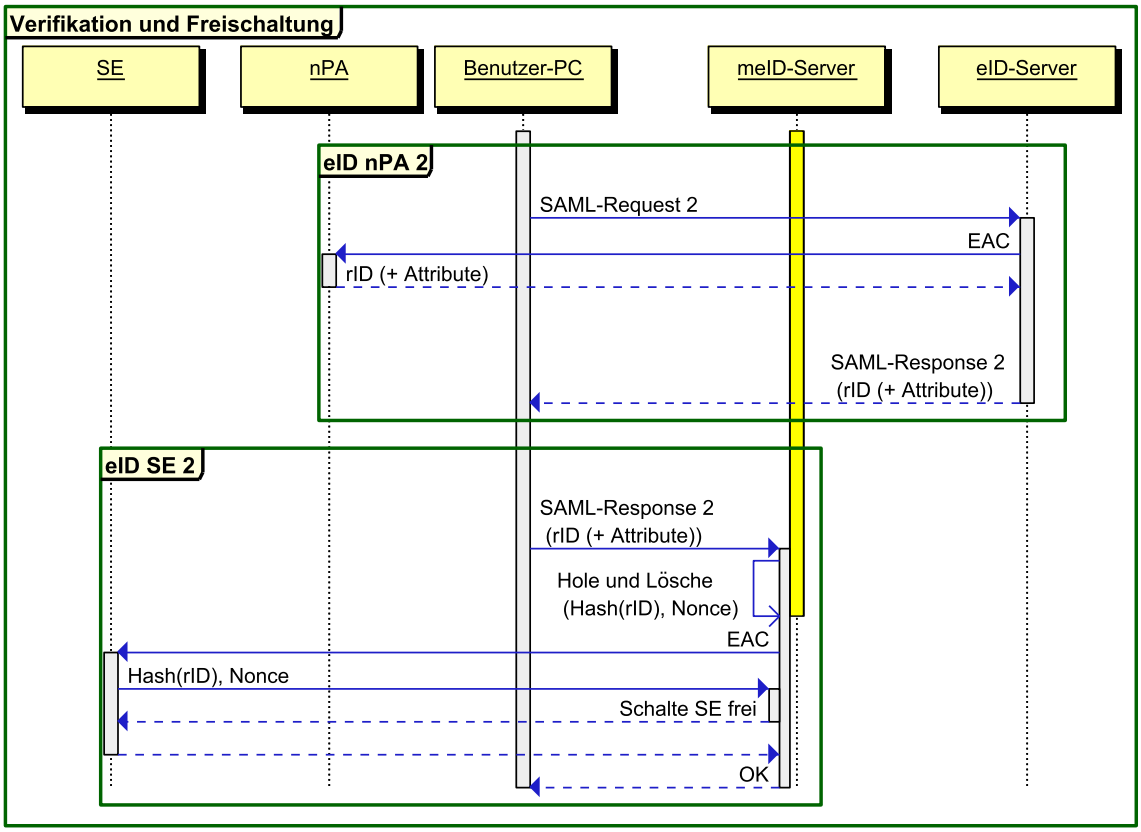

Abbildung 5.8.: Verifikation mit EAC

(a) Der eID-Server erhält den am Ende der Personalisierungsphase vom meID-Server erstellten *SAML-Request 2*.

(b) Der anschließende Auslesevorgang unterscheidet sich von dem Auslesevorgang der Personalisierung in dem verwendeten CV-Zertifikat und der Anzahl der ausgelesenen Attribute. Im CV-Zertifikat wird ein anderer Eigentümer und Auslesegrund angegeben. Wichtig ist, dass sich beide Zertifikate im selben Sektor befinden, damit die ausgelesene rID dieselbe ist. Neben der rID können weitere Attribute ausgelesen werden, die später bei der Ausstellung der Identitätstoken verwendet werden.

(c) Die *SAML-Response 2* wird vom eID-Server an den meID-Server gesendet.

(d) Der meID-Server hasht die empfangene rID und sucht in der Datenbank nach der zugehörigen Nonce. Findet er den entsprechenden Eintrag, wird er innerhalb einer atomaren Aktion gleichzeitig gelöscht. Somit ist sichergestellt, dass nur ein Thread die gehashte rID und Nonce aus der Datenbank abruft. Ist der Eintrag nicht vorhanden, bricht der Server den Vorgang ab.

(e) Wie in der Personalisierungsphase wird eine mit EAC gesicherte Verbindung zwischen meID-Server und Secure Element aufgebaut.

(f) Über diese Verbindung sendet das SE die gehashte rID und die Nonce an den meID-Server.

(g) Stimmen die gespeicherte und die übertragene gehashte rID und Nonce überein, sendet der meID-Server ein Freischaltkommando an das SE.

(h) Nach der Freischaltung kann die Ausstellung der Identitätstoken durchgeführt werden.

Der nPA als Datenquelle

Der Identifier, mit dem das Secure Element personalisiert wird, muss aus einer authentischen Datenquelle stammen, die nur mit Einwilligung des Benutzers ausgelesen werden kann. Des Weiteren muss er vertrauenswürdig zu dem meID-Server gelangen. Im vorliegenden Fall stellt der nPA die Datenquelle dar und der Identifier wird durch die rID repräsentiert. Die Verwendung des nPA als Datenquelle hat folgende Gründe:

- Für die Authentizität der ausgelesenen Daten bürgt der eID-Server.
- Die Authentifizierung für das Lesen der Daten aus dem nPA und das Schreiben in das SE wird mit der EAC beide Male durch dasselbe Protokoll realisiert.
- Die rID wird eindeutig durch die nPA-Diensteanbieter-Kombination bestimmt und wahrt zusätzlich die Anonymität des Benutzers.
- Neben der rID bietet der nPA weitere Attribute, wie Name und Anschrift, aus denen später die Identitätstoken gebildet werden können.

Weitere Datenquellen Es können auch weitere Datenquellen verwendet werden. Dabei muss nur sichergestellt werden, dass folgende, im Kapitel 3.2 genannten Annahmen, auch weiterhin zutreffen. Die Datenquelle kann nur mit Hilfe des Benutzers verwendet werden (Annahme A_1) und die Attribute müssen vertraulich und integer zum Token Issuer gelangen (Annahme A_9).

CV-Zertifikate für den nPA

Für die Personalisierung und die Verifikation müssen unterschiedliche CV-Zertifikate verwendet werden, in denen ein jeweils anderer Eigentümer und Auslesegrund angegeben sein muss. Somit kann der Benutzer über das Kartenleserdisplay sehen, in welcher Phase der Personalisierung er sich gerade befindet. Dies ist wichtig, um eine bestimmte Form der Personalisierungsumleitung zu verhindern.

PACE mit dem SE

Abbildung 5.9.: PIN-Eingabe für das SE eines mobilen Endgeräts auf einem Standardleser

Der Zweck und die Funktionsweise von PACE wurden in Abschnitt 2.3.1 betrachtet. Die für PACE benötigte PIN kann auf verschiedene Arten vorher festgelegt werden. So kann sie z. B. der Benutzer während der Appletinstallation festlegen. Da die Eingabe der PIN dabei über das Endgerät erfolgt, kann Schadsoftware auf dem Endgerät sie bei dem Vorgang abgreifen. Ebenfalls könnte das Applet während der Installation eine PIN wählen, die dann auf dem Endgerät angezeigt wird. In diesem Fall kann Schadsoftware die PIN von der Anzeige abgreifen. Die Eingabe der SE-PIN während der Personalisierung kann sich als unhandlich erweisen, sofern ein Standard- oder Komfortleser verwendet wird. Diese generieren das Feld auf ihrer Unterseite, sodass der Leser während der Eingabe auf dem Endgerät aufliegt. Da der Leser sowie die meisten Endgeräte über eine relativ glatte Oberfläche verfügen, kann es somit passieren, dass die Geräte während der Eingabe verrutschen und die Verbindung abbricht. Die Eingabe der SE-PIN während beide Geräte in der Hand gehalten werden, ist in Abbildung 5.9 zu sehen.

meID-Server

Der meID-Server befindet sich in zwei unterschiedlichen PKIs. Im deutschen eID-System ist er ein Diensteanbieter, der über den eID-Server die Ausweisdaten erhält. Dort besitzt er zwei Berechtigungszertifikate, die sich im selben Sektor befinden. Das erste benötigt er,

um die nPA-Daten für die Personalisierung auszulesen, das zweite wird für die Verifikation benötigt. Das SE befindet sich hingegen in einer anderen PKI. Für das SE besitzt er ein CV-Zertifikat mit Schreibberechtigung und eines mit Leseberechtigung für die rID. Sofern die Ausstellung der Identitätstoken mit der Personalisierung verbunden wird, muss der meID-Server über eine sichere Verbindung zum Issuer verfügen. Über diese Verbindung werden die nPA Attribute übertragen, die im Token kodiert werden.

Weitere Personalisierungsattribute

Die Attribute, die der Personalisierungstoken enthält, werden an dieser Stelle nicht festgelegt. Sollen später beim Issuing nur Token aus dem neuen Personalausweis abgeleitet werden, reicht die RI als einziger Identifizierer. Sollen auch andere Token, wie zum Beispiel ein elektronischer Führerschein, unterstützt werden, empfiehlt es sich, zusätzlich Name und Adresse mit zu personalisieren. Der temporäre Eintrag in der Datenbank des meID-Servers sollte hingegen nur die gehashte rID und die Nonce beinhalten.

5.2.3. Sicherheitsbetrachtung

Bei der Personalisierung und anschließender Verifikation handelt es sich um eine besonders kritische Phase, da hier die Zuordnung des SE zum Benutzer stattfindet. Die Ausstellung der Identitätstoken baut auf der Zuordnung auf. Wird das falsche SE personalisiert, erhält auch das falsche SE die Token. Für die eID-Funktion des neuen Personalausweises existieren bereits Schutzprofile des BMI [9, 16], sodass die Sicherheitsbetrachtung für *eID nPA 1* und *eID nPA 2* bei der Personalisierung bzw. Verifikation entfallen. Zum Großteil können die Schutzprofile auch auf *eID SE 1* und *eID SE 2* angewendet werden, da dabei ein Teil der eCard-API und insbesondere die EAC verwendet werden. Allerdings wird in den Schutzprofilen angenommen, dass Malware die Plattform, also den Benutzer-PC zusammen mit der eID-Anwendung, nicht manipuliert hat. Diese Annahme trifft bei der vorliegenden Arbeit hingegen nicht zu. Nichtsdestotrotz wird davon ausgegangen, dass die Attribute sicher vom nPA zum meID-Server (und nach Annahme A_9 auch zum Token Issuer) gelangen. Des Weiteren wird die EAC und der damit geschützte Übertragungskanal als sicher betrachtet. Alle weiteren Verbindungen, insbesondere die HTTP-Verbindungen des Benutzer-PCs sind angreifbar. Dabei müssen insbesondere der Empfang und Versand der SAML-Request und SAML-Response Nachrichten betrachtet werden.

Mitlesen der Attribute (*** / Wird verhindert)

Beschreibung Werden dem Benutzer Attribute auf der Seite des meID-Servers angezeigt, kann der Angreifer diese zum Beispiel über Malware auf dem PC mitlesen. Werden bei der Personalisierung und bei der Verifikation nur die rID aus dem nPA ausgelesen, ist der Angriff weniger schwerwiegend.

Einstufung Durch den Angriff werden die Attribute kopiert, sodass ihre Vertraulichkeit gefährdet ist und der Angriff als **schwerwiegend** eingestuft wird. Des Weiteren kann der Angriff als Vorbereitung des Identitätdiebstahls verwendet werden, wenn der Angreifer es schafft, den so ausgelesenen Identifizierer in ein eigenes SE zu spielen, sodass dieser Angriff ein **leichtes** Gefährdungspotenzial besitzt.

Mögliche Gegenmaßnahmen Der meID-Server sollte nach dem Erhalt der SAML-Response höchstens die Attributtypen auf der Webseite anzeigen.

Wiedereinspielen der SAML-Response 1 (** / Wird verhindert)

Beschreibung Der Angreifer kopiert zum Beispiel mit Hilfe von Malware auf dem Benutzer-PC die SAML-Response 1 und sendet sie ebenfalls an den meID-Server. Im Erfolgsfall wird sein SE und das SE des Opfers mit den gleichen Daten personalisiert.

Einstufung Der Angriff stellt die Grundlage für den Identitätsdiebstahl dar. Da anschließend noch die Verifikation durchlaufen muss, wird der Angriff als **mittel** eingestuft. Die Authentizität des Benutzers wird angegriffen.

Mögliche Gegenmaßnahmen Eine SAML-Response bezieht sich immer auf genau einen SAML-Request indem im *ResponseTo* Attribut die ID des entsprechenden SAML-Request steht. Der meID-Server sollte eine Liste mit IDs unbeantworteter SAML-Requests führen, bei der regelmäßig veraltete Einträge gelöscht werden. Erhält er eine SAML-Response, überprüft er, ob sie sich auf eine ID in der Liste bezieht. Im positiven Fall löscht er den Eintrag und fährt weiter fort. Im negativen Fall bricht er den Vorgang ab. Somit kann sichergestellt werden, dass eine SAML-Response nicht mehrmals verwendet werden kann. Sofern der Angreifer die Antwort vor dem Benutzer an den meID-Server sendet und der Vorgang somit für den Benutzer abgebrochen wird, hat der Angriff Ähnlichkeiten mit der *Umleiten der SAML-Response 1*.

Umleiten der SAML-Response 1 (** / Wird nicht verhindert)

Beschreibung Der Angreifer kopiert die SAML-Response 1 und sorgt gleichzeitig dafür, dass der Benutzer-PC sie nicht zum meID-Server schickt. So kann er den Benutzer zum Beispiel auf eine extra präparierte Seite weiterleiten, auf der er dazu aufgefordert wird, den Personalisierungsvorgang noch einmal neu zu beginnen.

Einstufung Der Angriff stellt die Grundlage für den Identitätsdiebstahl dar. Da anschließend noch die Verifikation durchlaufen muss, wird der Angriff als **mittel** eingestuft. Die Authentizität des Benutzers wird angegriffen.

Mögliche Gegenmaßnahmen Der Angriff lässt sich nicht definitiv verhindern. Einen gewissen Schutz bietet die Überprüfung, ob die IP des Benutzer-PCs von Beginn der Personalisierung bis zum Ende gleich geblieben ist. Diese Überprüfung ist jedoch unwirksam, wenn der Angreifer den Benutzer-PC als Proxy missbraucht.

Kopieren der Personalisierung (** / Wird nicht verhindert)

Beschreibung Der Angreifer kopiert den PSK und den Session-Identifier mit denen der TLS-PSK-RSA gesicherte Kanal zwischen der meID-Anwendung auf dem PC und dem meID-Server aufgebaut wird. Mit den Daten baut er ebenfalls solch eine Verbindung auf, über die sein SE personalisiert wird.

Einstufung Der Angriff stellt die Grundlage für den Identitätsdiebstahl dar. Da anschließend noch die Verifikation durchlaufen muss, wird der Angriff als **mittel** eingestuft. Die Authentizität des Benutzers wird angegriffen, des Weiteren wird die Vertraulichkeit des PSK und des Session-Identifiers angegriffen.

Mögliche Gegenmaßnahmen Der meID-Server sollte nur eine Verbindung mit dem selben PSK innerhalb des Zeitraums, in dem eine SAML-Response gültig ist, zulassen. Baut der Angreifer die TLS-PSK-RSA Verbindung vor dem Benutzer auf, kann er trotzdem sein SE personalisieren. Bei dem Benutzer wird dafür dann eine Fehlermeldung angezeigt.

Vortäuschen einer erfolgreichen Personalisierung (*** / Wird verhindert)

Beschreibung Der Angreifer versucht dem Benutzer eine erfolgreiche Personalisierung vorzutäuschen, obwohl als Folge von einem der drei vorherigen Angriffen nur das SE des Angreifers personalisiert wurde. Anschließend möchte der Benutzer sein SE freischalten, wobei der Angreifer wiederum die *SAML-Response 2* abfängt und sein SE freischaltet.

Einstufung Der Angriff hat den Identitätsdiebstahl als direkte Folge und wird somit als **schwer** eingestuft. Es wird die Authentizität des Benutzers angegriffen.

Mögliche Gegenmaßnahmen Dem Benutzer muss auf dem Kartenleserdisplay der erfolgreiche oder erfolglose Ablauf der EAC angezeigt werden. Verläuft die EAC korrekt, kann er sich sicher sein, dass sein SE personalisiert wird. Im Fehlerfall wird angenommen, dass der Benutzer die Verifikations- und Freischaltungsphase nicht einleitet, sondern von vorne mit dem Personalisierungsprozess beginnt. Dabei wird vom meID-Server eine neue Nonce generiert, die von der Nonce im SE des Angreifers abweicht, sodass sein SE nicht mehr freigeschaltet werden kann.

Vortäuschen einer erneuten Personalisierung (*** / Wird verhindert)

Beschreibung Der Angreifer versucht dem Benutzer eine erneute Personalisierung vorzutäuschen, nachdem nur das SE des Angreifers personalisiert wurde. Der Benutzer glaubt, sich in der Personalisierungsphase zu befinden, ist jedoch bereits in der Verifikations- und Freischaltphase und schaltet somit das SE des Angreifers frei.

Einstufung Der Angriff hat den Identitätsdiebstahl als direkte Folge und wird somit als **schwer** eingestuft. Es wird die Authentizität des Benutzers angegriffen.

Mögliche Gegenmaßnahmen Nach Annahme A_2 wird dem Benutzer der Auslesegrund bzw. die auslesende Partei auf einer vertrauenswürdigen Anzeige angezeigt. Im vorliegenden Fall wird auf dem Kartenleserdisplay der Eigentümer des Berechtigungszertifikats angezeigt. Für die Personalisierung und die Verifikation müssen zwei unterschiedliche Berechtigungszertifikate verwendet werden, die für unterschiedliche Dienste ausgestellt werden. Dies kann zum Beispiel einmal ein „Personalisierungsdienst“ und einmal ein „Verifikationsdienst“ sein. Somit kann der Benutzer über das Kartenleserdisplay die beiden Phasen voneinander unterscheiden und einen derartigen Angriff erkennen.

Kopieren oder Umleiten der SAML-Response 2 (*** / Wird verhindert)

Beschreibung Der Angreifer kopiert zum Beispiel mit Hilfe von Malware auf dem Benutzer-PC die SAML-Response 2 oder leitet sie direkt an den meID-Server. Im Erfolgsfall wird sein SE verifiziert und freigeschaltet.

Einstufung Der Angriff führt direkt zum Identitätsdiebstahl, weshalb die Gefährdung als **schwer** eingestuft wird. Die Authentizität des Benutzers wird angegriffen.

Mögliche Gegenmaßnahmen Selbst wenn es dem Angreifer gelungen ist, in einer anderen Session sein SE mit der rID personalisieren zu lassen, verfügt es nicht über die korrekte Nonce. Somit schlägt die Verifikation fehl. Der Fall, dass der Angriff in derselben Session durchgeführt wird, wird in „Vortäuschen einer erfolgreichen Personalisierung“ und „Vortäuschen einer erneuten Personalisierung“ betrachtet.

Fälschung des meID-Servers (** / Verhinderbar)

Beschreibung Ein Angreifer kann die Personalisierung des Benutzer-SE simulieren, indem er ein CV-Zertifikat mit gleicher Certificatedescription und gleichen Berechtigungen zum SE schickt, die Verbindung jedoch vor der TA abbricht (andernfalls würde die TA wegen des fehlenden Schlüsselmaterials scheitern). Da PACE zuvor erfolgreich durchgelaufen ist, denkt der Benutzer, dass sein SE personalisiert wurde.

Einstufung Der Angriff stellt eine Grundlage für den Identitätsdiebstahl dar. Da noch weitere Angriffe benötigt werden, wird der Angriff als **mittel** eingestuft.

Mögliche Gegenmaßnahmen Derzeit zeigen Kartenleser nur einen erfolgreichen PACE-Durchlauf auf ihrem Display an. Sofern diese Erfolgsmeldung erst nach der TA angezeigt wird, könnte der Benutzer den Angriff bemerken und die anschließende Verifikation nicht durchführen.

Relay Angriff (*** / Wird nicht verhindert) ⚠

Beschreibung Der Kartenleser kommuniziert über die kontaktlose Schnittstelle mit dem SE. Dafür muss sich der NFC-Chip des mobilen Endgerätes im Card-Emulation-Modus befinden. Im Normalfall kann davon ausgegangen werden, dass die Verbindung im SE terminiert. In [43] wird hingegen ein sogenanntes Software Secure Element vorgestellt, bei dem die Verbindung in einer Anwendung innerhalb des mobilen Endgeräts terminiert. Besitzt ein Angreifer Malware auf dem Endgerät, kann er darüber die Verbindung in ein eigenes entferntes Secure Element weiterleiten, welches personalisisert und anschließend auch freigeschaltet wird, ohne dass der Benutzer dies merkt.

Einstufung Der Angriff führt direkt zum Identitätsdiebstahl, sodass die Gefährdung als **schwer** eingestuft wird.

Mögliche Gegenmaßnahmen In der BlackBerry Plattform kann die UID einer emulierten Karte wegen Sicherheitsbedenken nicht selber festgelegt werden[9]. Eine Möglichkeit besteht nun darin, nur Secure Elements mit bestimmten UIDs zuzulassen. Allerdings ist unklar, inwiefern bei Android die UID frei gewählt werden kann. Wirksame Gegenmaßnahmen erfordern eine Anpassung des mobilen Endgeräts:

- Die Software Card Emulation wird im NFC-Chip komplett deaktiviert. Das Vorgehen ist wenig sinnvoll, da es für viele nicht sicherheitskritische Anwendungen durchaus Sinn macht, eine Chipkarte mit einer normalen Endgerätanwendung zu emulieren.
- Die Software Card Emulation wird mit einer TEE kombiniert, sodass nur vom TEE-Inhaber zugelassene Applikationen eine Karte emulieren können. Dafür muss jedoch ebenfalls der NFC-Chip angepasst werden.

Kreuzweise Umleitung mit einem fremden nPA (*** / Wird nicht verhindert) ⚠

Beschreibung Für den Angriff benötigt der Angreifer Zugriff auf einen nPA (*nPA'*) dessen PIN er kennt. Dies kann entweder sein eigener oder der eines zweiten entfernten Opfers sein, dessen nPA auf einem Basisleser liegt. Wie in Abbildung 5.10 zu sehen ist, leitet der Angreifer die ausgelesenen Daten kreuzweise um. In der Personalisierungsphase werden somit die Daten von *nPA'* in das SE des Benutzers geschrieben und umgekehrt. Da die Personalisierungsphase somit beim Benutzer nicht fehlschlägt, führt er die Verifikations- und Freischaltungsphase durch. In dieser wird das SE des Angreifers freigeschaltet. Der Angreifer muss in der zweiten Phase nicht zwingend noch einmal *nPA'* auslesen. Dies kann aber sinnvoll für ihn sein, wenn es sich bei *nPA'* nicht um seinen eigenen nPA handelt und die zweite Phase beim Benutzer nicht abbrechen soll.

[9] http://supportforums.blackberry.com/t5/Java-Development/UID-for-NFC-Mifare-Tag-emulation/m-p/1575809

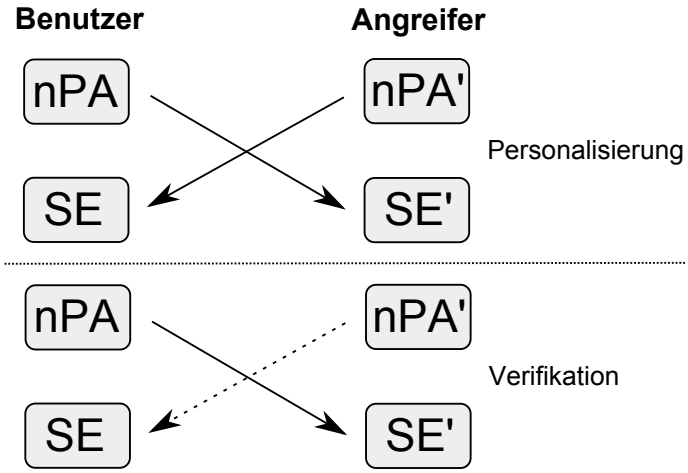

Abbildung 5.10.: Kreuzweise Umleitung

Einstufung Der Angriff führt direkt zum Identitätsdiebstahl, weswegen die Gefährdung als **schwer** eingestuft wird. Die Authentizität des Benutzers wird angegriffen.

Mögliche Gegenmaßnahmen Eine sinnvolle Gegenmaßnahme ist äußert schwierig zu finden.

- Der meID-Server muss darauf achten, dass die SAML-Response von dem selben PC kommt, an den er den SAML-Request versendet hat. Die Zuordnung kann zum Beispiel über die IP-Adresse oder mit einem Browsercookie erfolgen. Allerdings kann diese Zuordnung durch spezialisierte Schadsoftware ausgehebelt werden.
- Für den Angriff müssen auf jeden Fall Daten aus einem echten nPA in das SE des Benutzers übertragen werden. Sofern der Benutzer den Angriff bemerkt, könnte ein spezieller Dienst die so personalisierten Daten aus seinem SE auslesen und darüber den Inhaber von *nPA'* ermitteln. Dafür muss allerdings die Personalisierung nicht mit der rID sondern mit z. B. der Anschrift durchgeführt werden.

5.2.4. Aktuelle Grenzen

Kreuzweise Umleitung

Die kreuzweise Umleitung stellt eine ernst zu nehmende Bedrohung dar, die derzeit nicht effektiv abgewehrt werden kann. Da der Angreifer freien Zugriff auf *nPA'* besitzt, bringt auch das Hintereinanderschalten mehrerer Verifikationsphasen keinen Sicherheitsvorteil. Sofern der Angreifer seinen eigenen nPA benutzt hat, könnte er hinterher über die personalisierten Daten ermittelt werden. Voraussetzung dafür ist allerdings, dass er keinen

Abbildung 5.11.: Schreibzugriff auf DG21

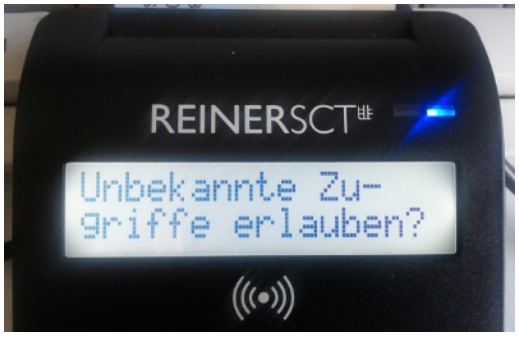

Abbildung 5.12.: Zugriff auf DG13-16

Zugriff auf einen entfernten nPA besitzt. Dies kann zum einen dadurch realisiert werden, dass Benutzer ihren nPA nur für die Dauer einer Authentisierung auf dem Kartenleser liegen lassen. Somit sinkt die Wahrscheinlichkeit, dass der Angreifer während der Personalisierung Zugriff auf einen unbeaufsichtigten nPA, der auf einem Basisleser liegt, hat. Des Weiteren wird empfohlen, nur noch Standard- und Komfortleser zu verwenden. Da bei PACE die PIN auf deren Tastatur eingegeben werden muss, kann der Vorgang nicht mehr von einem entfernten Angreifer durchgeführt werden.

Darstellung der Schreibberechtigungen

Der meID-Server sendet ein CV-Zertifikat zum SE. Das Zertifikat enthält neben dem Anbieternamen und Zugriffsgrund auch die Zugriffsberechtigungen, die auf dem Kartenleserdisplay angezeigt werden. Die Berechtigungen sind in der TR-03110 spezifiziert und werden in einzelne Datengruppen (DG) unterteilt. Für Schreibvorgänge sind die DG 17 bis 21 vorgesehen, wobei nur DG 21 noch nicht benutzt wird[10]. Enthält das Zertifikat die DG 21 Berechtigung, wird das auf dem Kartenleserdisplay der in Abbildung 5.11 zu sehende Text „Schreibzugriff Datengruppe 21“ angezeigt. Des Weiteren existieren vier als RFU[11] gekennzeichnete Felder DG 13 bis 16, deren Verwendung die in Abbildung 5.12 zu sehende Meldung „Unbekannte Zugriffe erlauben?“ hervorruft. Fraglich ist, ob ein Benutzer solchen Meldungen zustimmen würde. Für eine Änderung müsste die Firmware der Kartenleser angepasst werden, was meiner Meinung zwar einen geringen Programmieraufwand bedeuten würde aber trotzdem unrealistisch ist, da der Kartenleser neu zertifiziert werden muss.

[10] Die Datengruppen 17 und 18 sind zur Adressänderung und 19 und 20 zur Änderung der Aufenthaltsgenehmigung vorgesehen.
[11] Reserved for future use

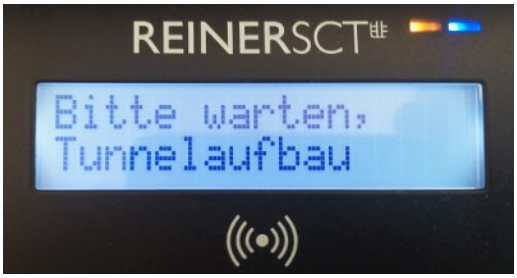

Abbildung 5.13.: Tunnelaufbau

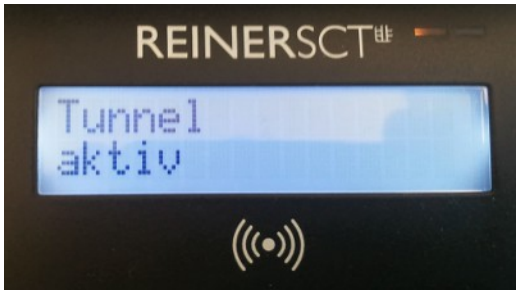

Abbildung 5.14.: Tunnel aktiv

Darstellung der sicheren Übertragung auf dem Leserdisplay

Der Ablauf von PACE wird bereits auf dem Kartenleserdisplay angezeigt, wie in den Abbildungen 5.13 und 5.14 zu sehen ist. Ein Verbindungsabbruch oder Erfolg der TA und CA wird hingegen nicht auf dem Leserdisplay angezeigt. Für das System neuer Personalausweis spielt diese eine eher untergeordnete Rolle, da der Benutzer spätestens vom Diensteanbieter über einen fehlerhaften Durchlauf informiert wird, da dieser keine Attribute erhalten hat. Bei der Personalisierung eines SE erhält der Benutzer jedoch keine Rückmeldung von seinem SE. Ein Angreifer kann somit eine erfolgreiche Personalisierung des Benutzer-SE fälschen, indem er einen meID-Server nachahmt und ein gefälschtes CV-Zertifikat zum SE sendet. Das Protokoll bricht dann zwar spätestens bei der TA ab, dies wird jedoch auf Grund der fehlenden Rückmeldung nicht vom Benutzer gemerkt. Dabei hat ein Kartenleser durchaus die Möglichkeit, den erfolgreichen Ablauf der EAC an Hand der Kommandobytes und Returncodes zu verifizieren.

Es wird empfohlen, die Anzeige „Bitte warten, Tunnelaufbau“ (Abbildung 5.13) bis zum Ende der EAC zu verlängern und erst dann einen aktiven Tunnel anzuzeigen (Abbildung 5.14). Sofern die EAC abgebrochen wird, sollte dies ebenfalls auf dem Display angezeigt werden.

5.3. Ausstellung der Identitätstoken

Bei dem Ausstellen der Identitätstoken, auch *Issuing* genannt, werden die Identitätstoken auf dem Gerät gespeichert, mit denen sich der Benutzer später authentisieren kann. Das Issuing baut auf die Personalisierung und somit auf die bereits bestehende Zuordnung zwischen Datenquelle und SE auf. Es wird in dieser Arbeit weniger detailliert beschrieben, als die Personalisierung. Unter anderem wird von dem in [39] beschriebenen U-Prove Issuingprotokoll abstrahiert, da dieses nicht Bestandteil der Arbeit ist.

(a) Der Benutzer möchte sich eine bestimmte Anzahl neuer Identitätstoken ableiten lassen und wählt den entsprechenden Menüpunkt in der UI-Applikation.

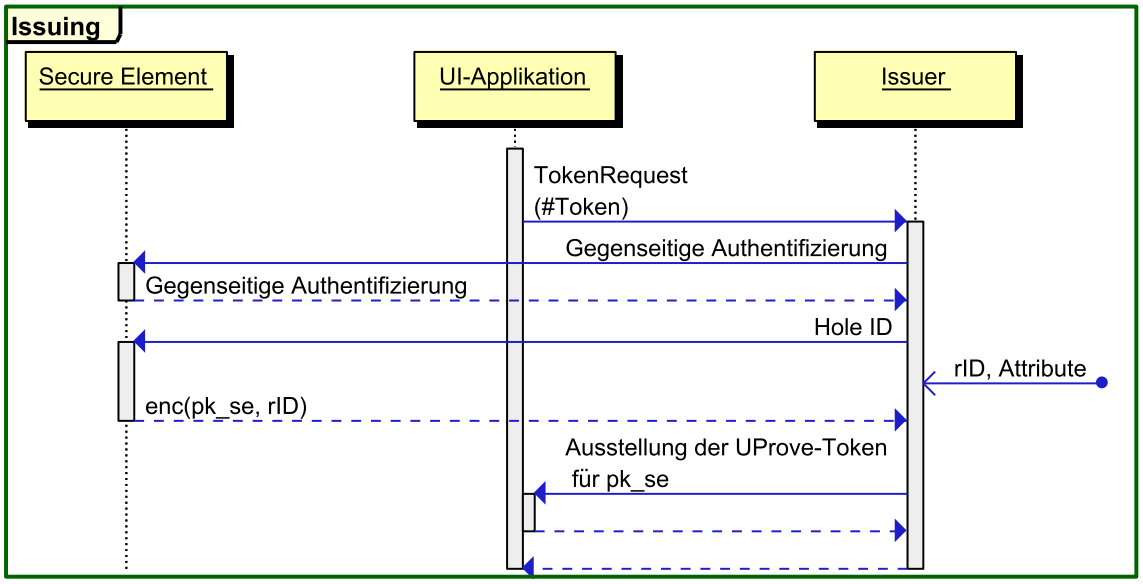

Abbildung 5.15.: Issuing

(b) Die Applikation sendet einen TokenRequest zusammen mit der gewünschten Anzahl an Token zum Issuer.

(c) Der Issuer führt über die UI-Applikation eine gegenseitige Authentifizierung mit dem Secure Element durch und baut eine mit Secure Messaging geschützte Verbindung auf, bei der die Nachrichten verschlüsselt und signiert werden. Die Authentifizierung und der Verbindungsaufbau können mit symmetrischer oder asymmetrischer Kryptographie erfolgen.

(d) Über den sicheren Kanal sendet das SE seinen öffentlichen Schlüssel für die Tokenausstellung und den gespeicherten Identifizierer, in dem Fall die rID.

(e) Währenddessen erhält der Issuer vom meID-Server die rID und die Attribute für die Tokenausstellung. Dies kann zum Beispiel zum Ende der Freischaltphase in der Personalisierung (siehe dazu Abschnitt 5.2.2) geschehen. Der Benutzer kann sich aber auch nur für die Tokenausstellung gegenüber dem meID-Server authentifiziert haben.

(f) Der Issuer ordnet mit Hilfe der rID das SE und die Attribute zueinander zu.

(g) Anschließend werden bei dem Token Issuing Protokoll von U-Prove die Token ausgestellt.

5.3.1. Personalisierung und Issuing

Der Ausstellungsvorgang kann mit der Personalisierung (genauer gesagt der Freischaltung) verbunden werden. Dabei werden die auszustellenden Attribute zum Anfang der

Verifikations- und Freischaltphase ebenfalls aus der Datenquelle ausgelesen und für die Sitzungsdauer gespeichert[12] Wurde das SE frei geschaltet, sendet der meID-Server die Attribute und die rID über eine direkte Verbindung zum Issuer und löscht sie lokal. Gleichzeitig drückt der Benutzer in der UI-Applikation einen Issuingbutton. Daraufhin verbindet sich die UI-Applikation mit dem Issuer und der Issuing-Prozess beginnt. Der Vorteil besteht darin, dass die bereits für die Personalisierung ausgelesenen Attribute für das Issuing weiterverwendet werden können. Nichtsdestotrotz sollten nachfolgende Ausstellungsvorgänge immer mit frisch aus dem nPA ausgelesenen Attributen durchgeführt werden, um die Aktualität der Daten zu gewährleisten.

5.3.2. Sicherheitsbetrachtung

Unbefugtes Auslesen (*** / Wird verhindert)

Beschreibung Der Angreifer gibt sich mit Hilfe von Malware auf dem Smartphone gegenüber dem SE als Issuer aus und erhält den öffentlichen Schlüssel pk_se und die *rID*.

Einstufung Die Vertraulichkeit des öffentlichen Schlüssels pk_se und der *rID* ist gefährdet und die Authentizität des Issuers wird angegriffen. Da die *rID* ein nPA-Attribut darstellt, ist die Gefährdung **schwer**.

Mögliche Gegenmaßnahmen Durch die gegenseitige Authentifizierung muss sichergestellt werden, dass nur vertrauenswürdige Akteure die rID erhalten.

Replay Angriff (*** / Wird verhindert)

Beschreibung Der Angreifer schneidet die Kommunikation mit und sendet die Nachrichten an sein eigenes SE. Die Identitätstoken werden folglich auch in sein SE gespielt.

Einstufung Der Angriff führt direkt zum Identitätsdiebstahl, sodass die Gefährdung als **schwer** eingestuft wird.

Mögliche Gegenmaßnahmen Die gegenseitige Authentifizierung muss mit Hilfe eines Challenge-Response-Verfahrens gelöst werden, bei dem die Challenge zufällig von dem jeweiligen Akteur generiert wird. Die Verschlüsselungsschlüssel müssen dabei bei jedem Durchlauf zufällig neu generiert werden. Des Weiteren besitzt das Issuing-Protokoll von U-Prove von Hause aus einen Schutz gegen Replayattacken, da Prover und Issuer zu Beginn zufällige Werte bilden und somit die Frische der Ausstellung garantiert ist.

[12] Aus Datenschutzgründen sollte eine maximale Speicherdauer im einstelligen Minutenbereich gewählt werden.

6. Auswertung

6.1. Umsetzung der Zielstellung

Die Speicherung und Bereitstellung der Identitätstoken wurde bereits in den entsprechenden Kapiteln bewertet. Auch wurden die bekannten Grenzen genannt. Im aktuellen Abschnitt wird betrachtet, wie gut die Zielstellung aus Abschnitt 1.2 erfüllt werden konnte.

I. Bei der Übertragung muss die Vertraulichkeit und Integrität der Identitätstoken gewährleistet sein.

Für die Übertragung wird der Token Issuer als vertrauenswürdige Instanz verwendet. Die sichere Übertragung der Attribute von der Datenquelle zum Issuer ist nicht Teil der Arbeit und kann zum Beispiel mit der eID-Funktion des neuen Personalausweises realisiert werden. Das Issuing wird mit Secure Messaging und dem U-Prove Issuingprotokoll abgesichert.

II. Die Identitätstoken müssen in dem korrekten Gerät gespeichert werden.

Die Übertragung vom Issuer zum mobilen Endgerät baut auf der korrekten Personalisierung auf. Wurde das falsche SE personalisiert, werden auch die Token in das falsche SE übertragen. Die Zweiteilung der Personalisierung stellt einen Kompromiss zwischen Geschwindigkeit und Benutzerfreundlichkeit auf der einen und Sicherheit auf der anderen Seite dar. Dabei wird davon ausgegangen, dass sich der Benutzer aufmerksam die Meldungen auf dem Kartenleserdisplay durchliest, mit dem Ablauf vertraut ist und im Fehlerfall den Vorgang abbricht. Ein offenes Problem ist die kreuzweise Umleitung.

III. Es dürfen nur berechtigte Akteure in vorgegebener Weise auf den Datenspeicher zugreifen.

Durch die Nutzung eines SE ist die Vertraulichkeit und Integrität der Attribute auch nach der Übertragung sichergestellt. Die Attribute und ein Teil der privaten Tokenschlüssel werden durch das SE geschützt. Die Identitätstoken liegen hingegen ungeschützt im Dateisystem des Endgeräts und können von Schadsoftware kopiert werden. Für die Verwendung eines Tokens muss das SE mit einer PIN freigeschaltet werden. Im Gegensatz zu einer Chipkarte ist das SE ständig mit dem Gerät verbunden. Greift ein Angreifer die PIN ab, kann er, solange sich Token im Gerät befinden und er eine Verbindung zum Gerät hergestellt hat, die meID-Funktion missbrauchen.

IV. Das System sollte konform zu bestehenden Standards sein.

Bei dem Verfahren wurde, wenn möglich, auf bestehende Standards und Spezifikationen zurückgegriffen. Für die Verwaltung des SE wird die GlobalPlatform-Kartenspezifikation verwendet. Die Personalisierung orientiert sich an der eID-Funktion des nPA, ist aber ein selbst erfundenes Verfahren. Als Tokenformat werden U-Prove-Token verwendet. Auch ist in U-Prove spezifiziert, wie ein Secure Element in die Ausstellung und die Authentisierung einbezogen wird. Gleichzeitig wurde versucht, das Verfahren modular zu beschreiben, damit Teile davon ausgetauscht werden können. Der nPA als verwendete Datenquelle kann ebenso wie U-Prove als verwendetes ACS ausgetauscht werden, sofern die Alternativen, die genannten Anforderungen erfüllen. Als Nachteil stellt sich heraus, dass für die korrekte Umsetzung die Firmware eines Kartenlesers angepasst werden muss.

V. Das System sollte performant und leicht nutzbar sein.

Die gemeinsame Nutzung eines SE und des Dateisystems stellt einen guten Kompromiss zwischen Geschwindigkeit und Sicherheit dar. Eine PIN-Eingabe zur Authentifizierung ist bereits allgemein akzeptiert. Hingegen ist die Personalisierung aus Nutzersicht noch nicht optimal gelöst. Wegen der Aufteilung der Personalisierung müssen die Attribute zwei mal aus der Datenquelle ausgelesen werden. Geschieht das Auslesen ebenfalls über den Kartenleser, muss der Benutzer abwechselnd die Datenquelle, dann das Endgerät, dann wieder die Datenquelle und zum Schluss noch einmal das Endgerät an den Kartenleser halten. Die Benutzerfreundlichkeit ist somit eingeschränkt. Weiterhin ist es umständlich, auf einem Standard- oder Komfortleser eine PIN einzugeben, während er über die kontaktlose Schnittstelle mit einem mobilen Endgerät verbunden ist. Die Nutzung einer alternativen Schnittstelle mit etwas höherer Reichweite, wie zum Beispiel Bluetooth LE, wäre hier sicherlich von Vorteil. Alles in allem wird hierbei wieder deutlich, dass eine Erhöhung der Sicherheit meistens mit einer Verringerung der Nutzerfreundlichkeit einhergeht.

6.2. Schlusswort

Die sichere Bereitstellung von Identitätstoken auf einem mobilen Endgerät ist auch in einer unsicheren Umgebung möglich. Eine Aufteilung der Aufgabe in die Speicherung und die eigentliche Bereitstellung der Token erscheint sinnvoll. Die sichere Speicherung ist durch die Nutzung eines Secure Elements gut möglich. Dabei werden die Attribute sowie die privaten Tokenschlüssel durch das SE geschützt. Zur Verwendung der Token muss das SE durch eine PIN freigeschaltet werden. Die Schwachstelle liegt darin, dass die PIN-Eingabe von Schadsoftware abgegriffen werden kann. Ein Angreifer kann somit aus der Ferne die eID-Funktion für seine Zwecke benutzen. Abhilfe könnte hier eine hardwaregestützte Zugriffskontrolle auf das SE oder die Nutzung einer Trusted Execution Environment schaffen.

Bei der Übertragung der Identitätstoken muss darauf geachtet werden, dass die Daten in das korrekte Gerät geladen werden. Ein Umleitungsangriff kann jedoch nur schwer verhindert werden. Der Angreifer kann bereits die Installation des Applets im Secure Element umleiten, ohne dass der Benutzer dies merkt. Das ist umso erstaunlicher, als dass die Installation eines Applets in einem Secure Element bereits durch den GlobalPlatform Standard geregelt ist. Dieser stellt allerdings nur sicher, dass ein Applet in einem Secure Element installiert wird, jedoch nicht in welchem. Ursächlich dafür ist die fehlende vertrauenswürdige Ein- und Ausgabe eines Secure Elements, sowie dass das herausgegebene Secure Element unpersonalisiert ist. Die Personalisierung in einer unsicheren Umgebung stellt somit die eigentliche Herausforderung dar. Dies wird mittels eines zweiphasigen Verfahrens weitestgehend gelöst. In einer ersten Phase werden die Personalisierungsdaten aus einer vertrauenswürdigen Datenquelle ausgelesen und durch die EAC geschützt über einen Klasse 3 Kartenleser in das SE geschrieben. Dabei wird dem Benutzer auf dem Kartenleserdisplay eine Rückmeldung über den korrekten oder fehlerhaften Verlauf der Personalisierung gegeben. In einer anschließenden Verifikations- und Freischaltungsphase wird die Datenquelle wieder ausgelesen. Diesmal werden die im SE gespeicherten mit den ausgelesenen Daten verglichen. Stimmen diese überein, wird das SE freigeschaltet und kann erst ab dann zur Authentifizierung genutzt werden. Das Verfahren baut darauf auf, dass der Benutzer diese zweite Phase nicht ausführt, nachdem Unregelmäßigkeiten in der ersten Phase aufgetreten sind.

Für die Umsetzung des Verfahrens müssen der Standard- und Komfortleser angepasst werden. Es wird empfohlen, dass deren Display über einen fehlerhaften oder korrekten Durchlauf der EAC informiert. Außerdem sollte eine Datengruppe der CV-Zertifikatsberechtigungen für die SE-Personalisierung reserviert werden. Ein weitaus größeres Problem stellt die geringe Nutzerfreundlichkeit dar. Das zweifache Auslesen der Datenquelle und das komplizierte Hantieren mit dem Kartenleser und dem mobilen Endgerät dürfte sich zweifelsohne negativ auf die Akzeptanz auswirken. Auch ist das Verfahren gegenüber einer Art von Angriff, dem zweifachen Sessionhijacking, anfällig, wofür der Angreifer allerdings gewisse Voraussetzungen schaffen muss.

Trotz der Probleme kann das Verfahren als eine Grundlage für eine mobile eID-Implementierung angesehen werden, die auf Attribute aus einer authentischen Datenquelle aufbaut.

6.3. Ausblick

Der nächste konsequente Schritt ist die prototypische Umsetzung des vorgestellten Verfahrens. Mit dem Prototyp lässt sich dann überprüfen, wie die Benutzerfreundlichkeit und -akzeptanz erhöht werden kann.

Die Einbeziehung einer TEE im mobilen Endgerät kann beide Eigenschaften erhöhen. Durch das vertrauenswürdige Interface der TEE kann auf einen extra Kartenleser verzichtet werden. Gleichzeitig kann darüber eine Zugriffsbeschränkung für die Benutzung der

Token implementiert werden, sodass ein entfernter Angreifer nicht mehr per Malware die eID-Funktion verwenden kann. Ebenfalls ist denkbar, dass durch eine vertrauenswürdige Anwendung im TEE ganz auf das SE und dessen Personalisierung verzichtet werden kann. Über das Interface der TEE würde sich die Personalisierung auch um einiges einfacher gestalten.

Das vorgestellte Verfahren eignet sich auch für andere Datenquellen, etwa einem elektronischen Führerschein oder einem Betriebsausweis. Die Datenquelle muss nur den genannten Anforderungen entsprechen. Ebenfalls kann über eine Alternative zu U-Prove nachgedacht werden. Interessant dürfte dabei das System sein, welches derzeit bei ABC4Trust entwickelt wird. Dieses wird U-Prove sowie IDEMIX kapseln und gleichzeitig die Verwendung eines Secure Elements zulassen.

Die Personalisierung mit anschließender Verifikation kann auch für andere Bereiche, in denen Secure Elements in einer unsicheren Umgebung personalisiert werden müssen, angewendet werden. Ein Beispiel sind intelligente Stromzähler, sogenannte Smart Meter, die mit einem Secure Element ausgeliefert werden. Kauft sich ein Kunde ein neues Smart Meter oder wechselt er den Stromanbieter, muss das eingebaute Secure Element neupersonalisert werden.

A. Anhang

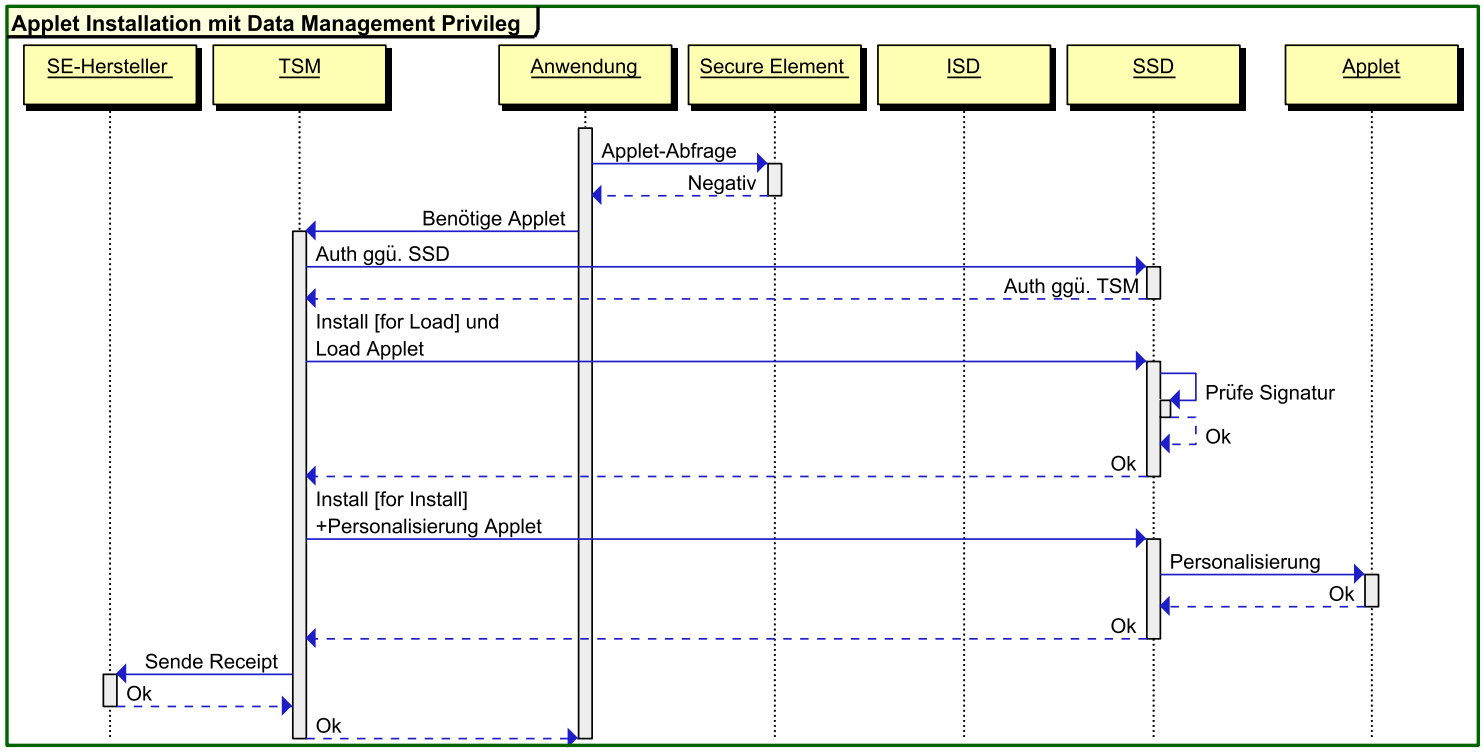

Abbildung A.1.: JC-Applet Installation mit DM

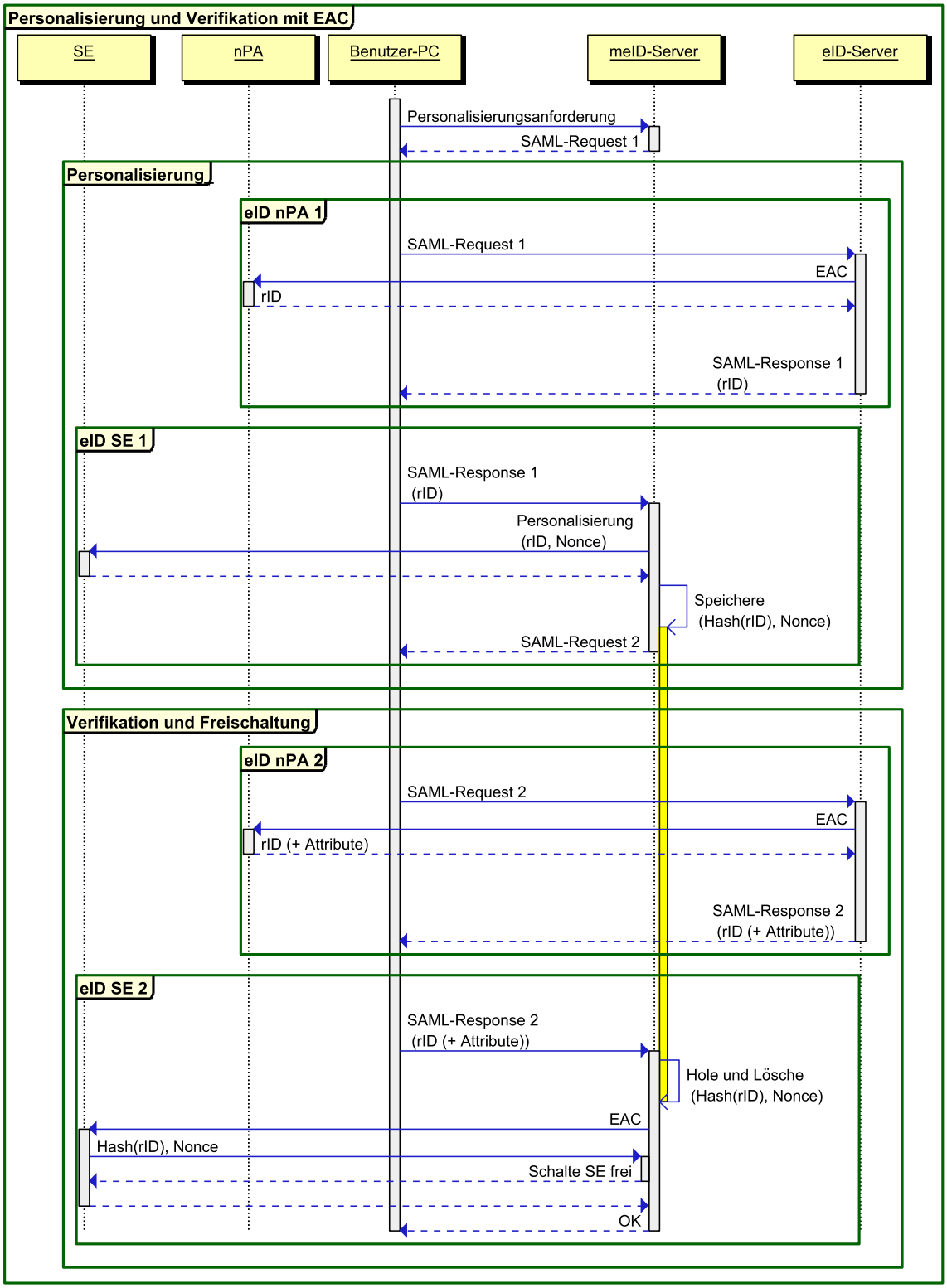

Abbildung A.2.: Personalisierung und Verifikation mit EAC

Literaturverzeichnis

[1] *Android.* http://www.android.com/,

[2] *AusweisApp.* https://www.ausweisapp.bund.de/pweb/index.do,

[3] *GlobalPlatform.* http://www.globalplatform.org,

[4] *Secure Element Evaluation Kit for the Android platform.* http://code.google.com/p/seek-for-android/,

[5] *SIMalliance - Open Mobile API.* http://www.simalliance.org/en/about/workgroups/open_mobile_api_working_group/,

[6] *Tor Project.* https://www.torproject.org/,

[7] *ISO 14443 - Identification cards - Contactless integrated circuit(s) cards - Proximity cards.* Februar 2001. – ISO/IEC

[8] *ABC4Trust - Project description.* https://abc4trust.eu/download/ABC4Trust-Project-Description.pdf, Mai 2009. – ABC4Trust

[9] *Common Criteria Protection Profile - Electronic Identity Card (ID_ Card PP).* https://www.bsi.bund.de/ContentBSI/Themen/Elekausweise/TRundSchutzprofile/TR_Schutzprofile/Schutzprofile.html, 2009. – Bundesministerium des Innern

[10] *Glossar und Begriffsdefinitionen.* https://www.bsi.bund.de/DE/Themen/weitereThemen/ITGrundschutzKataloge/Inhalt/Glossar/glossar_node.html, 2009. – Bundesamt für Sicherheit in der Informationstechnik

[11] *Specification of the Identity Mixer Cryptographic Library Version 2.3.0.* http://idemix.wordpress.com, März 2009. – IBM Research

[12] *ARD/ZDF-Onlinestudie.* http://www.ard-zdf-onlinestudie.de, 2009-2011

[13] *Nexus S Teardown.* http://www.ifixit.com/Teardown/Nexus-S-Teardown/4365/1, 2010. – ifixit

[14] *Technische Richtlinie TR-03110, Advanced Security Mechanisms for Machine Readable Travel Documents - Extended Access Control (EAC), Password Authenticated Connection Establishment (PACE) and Restricted Identification (RI), Version 2.05.* https://www.bsi.bund.de/cln_183/ContentBSI/Publikationen/TechnischeRichtlinien/tr03110/index_htm.html, Oktober 2010. – Bundesamt für Sicherheit in der Informationstechnik

[15] *Technische Richtlinie TR-03130, eID-Server Version 1.4.1.* https://www.bsi.bund.de/cln_174/ContentBSI/Publikationen/TechnischeRichtlinien/tr03130/tr-03130.html, Oktober 2010. – Bundesamt für Sicherheit in der Informationstechnik

[16] *Common Criteria Protection Profile - eID-Client based on eCard-API.* https://www.bsi.bund.de/ContentBSI/Themen/Elekausweise/TRundSchutzprofile/TR_Schutzprofile/Schutzprofile.html, 2011. – Bundesministerium des Innern

[17] *GlobalPlatform Card - Confidential Card Content Management - Card Specification v2.2 - Amendment A Version 1.0.1.* http://www.globalplatform.org/specificationscard.asp, Januar 2011. – GlobalPlatform

[18] *GlobalPlatform Card Specification Version 2.2.1 Public Release.* http://www.globalplatform.org/specificationscard.asp, Januar 2011. – GlobalPlatform

[19] *GlobalPlatform Device Technology - TEE System Architecture Version 1.0.* http://www.globalplatform.org/specificationsdevice.asp, Dezember 2011. – GlobalPlatform

[20] *GlobalPlatform System - Messaging Specification for Management of Mobile-NFC Services Version 1.0.* http://www.globalplatform.org/specificationssystems.asp, Februar 2011. – GlobalPlatform

[21] *Technische Richtlinie BSI TR-03119, Anforderungen an Chipkartenleser mit nPA Unterstützung Version 1.2.* https://www.bsi.bund.de/cln_183/ContentBSI/Publikationen/TechnischeRichtlinien/tr03112/index_htm.html, Mai 2011. – Bundesamt für Sicherheit in der Informationstechnik

[22] *GlobalPlatform Device Technology - Secure Element Access Control Version 1.0 .* http://www.globalplatform.org/specificationsdevice.asp, Mai 2012. – GlobalPlatform

[23] *Technische Richtlinie TR-03112, eCard-API-Framework Version 1.1.2.* https://www.bsi.bund.de/cln_183/ContentBSI/Publikationen/TechnischeRichtlinien/tr03112/index_htm.html, Februar 2012. – Bundesamt für Sicherheit in der Informationstechnik

[24] AL, S. C.: *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0.* http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf, März 2005. – OASIS SSTC

[25] BACKES, Michael ; GERLING, Sebastian ; HAMMER, Christian ; MAFFEI, Matteo ; STYP-REKOWSKY, Philipp von: *The Android Monitor - Real-time policy enforcement for third-party applications.* 2012

[26] BEILKE, Kristian: *Mobile eCard-API.* http://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2010-12/SAR-PR-2010-12_.pdf, Oktober 2010. – Humboldt-Universität zu Berlin

[27] Bu, Zheng ; Dirro, Toralv ; Greve, Paula ; Lin, Yichong ; Marcus, David ; Paget, François ; Schmugar, Craig ; Shah, Jimmy ; Sommer, Dan ; Szor, Peter ; Wosotowsky, Adam: *McAfee Threats Report: First Quarter 2012.* http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q1-2012.pdf, Mai 2012. – McAfee Labs

[28] Bugiel, Sven ; Davi, Lucas ; Dmitrienko, Alexandra ; Heuser, Stephan ; Sadeghi, Ahmad-Reza ; Shastry, Bhargava: *Practical and Lightweight Domain Isolation on Android.* http://www.informatik.tu-darmstadt.de/fileadmin/user_upload/Group_TRUST/PubsPDF/spsm18-bugiel.pdf, Oktober 2011. – Technische Universität Darmstadt, Fraunhofer SIT

[29] Chen, Zhiqun: *Java Card Technology for Smart Cards: Architecture and Programmer's Guide.* Addison-Wesley, 2004

[30] Edlund, Lasse: *Efficient U-Prove Implementation for Anonymous Credentials on Smart Cards.* http://wwwhome.ewi.utwente.nl/~mostowskiwi/papers/securecomm2011.pdf, 2011. – Radboud University Nijmegen

[31] Gartner: *Gartner Says Worldwide Smartphone Sales Soared in Fourth Quarter of 2011 With 47 Percent Growth.* http://www.gartner.com/it/page.jsp?id=1924314, Februar 2012

[32] Hansen, Christian: *A Framework for Identity and Privacy Management on Mobile Devices.* http://brage.bibsys.no/hia/handle/URN:NBN:no-bibsys_brage_15265, August 2010. – University of Agder

[33] Hyppönen, Konstantin: *An Open Mobile Identity Tool: An Architecture for Mobile Identity Management.* http://www.springerlink.com/content/hlup4k0v47775437/, 2008. – University of Kuopio

[34] Innern, Bundesministerium des: *Bildmaterial.* http://www.personalausweisportal.de/DE/Presse_Service/Bildmaterial/bildmaterial_node, 2010

[35] MittelstandsWiki: *Reiner SCT zeigt Kartenleser für Personalausweise.* http://www.mittelstandswiki.de/2011/02/cebit-2011-reiner-sct-zeigt-kartenleser-fur-personalausweise/, 2011

[36] Morgner, Frank ; Oepen, Dominik: *„Die gesamte Technik ist sicher": Besitz und Wissen: Relay-Angriffe auf den neuen Personalausweis.* http://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2010-13/SAR-PR-2010-13.pdf, 2010. – Humboldt Universtität zu Berlin

[37] Mostowski, Wojciech ; Vullers, Pim: *Secure and Confidential Applications on UICC.* http://www.nada.kth.se/utbildning/grukth/exjobb/rapportlistor/2009/rapporter09/edlund_lasse_09084.pdf, 2009. – KTH Computer Science and Communication

[38] Nohl, Karsten ; Melette, Luca: *GPRS Intercept: Wardriving your country.* http://events.ccc.de/camp/2011/Fahrplan/events/4504.en.html, 2011

[39] Paquin, Christian: *U-Prove Cryptographic Specification V1.1 - Draft Revision 1.* http://research.microsoft.com/pubs/166969/U-ProveCryptographicSpecificationV1.1.pdf, Februar 2011. – Microsoft Corporation

[40] Paquin, Christian: *U-Prove Technology Overview V1.1.* http://research.microsoft.com/pubs/166980/U-ProveTechnologyOverviewV1.1.pdf, Februar 2011. – Microsoft Corporation

[41] Persson, Christian: *Android-Updates kommen zu langsam.* http://heise.de/-1517381, April 2012. – Heise Zeitschriften Verlag

[42] Rankl, Wolfgang ; Effing, Wolfgang: *Handbuch der Chipkarten: Aufbau - Funktionsweise - Einsatz von Smart Cards.* Carl Hanser Verlag GmbH & CO. KG, 2008

[43] Roland, Michael: *Software Card Emulation in NFC-enabled Mobile Phones: Great Advantage or Security Nightmare?* http://www.medien.ifi.lmu.de/iwssi2012/papers/iwssi-spmu2012-roland.pdf, 2012. – University of Applied Sciences Upper Austria

Alle URLs wurden zuletzt am 07.10.2012 geprüft.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe. Weiterhin erkläre ich, eine Diplomarbeit in diesem Studiengebiet erstmalig einzureichen.

Berlin, den 29. Oktober 2012 .....

Einverständniserklärung

Ich erkläre hiermit mein Einverständnis, dass die vorliegende Arbeit in der Bibliothek des Institutes für Informatik der Humboldt-Universität ausgestellt werden darf.

Berlin, den 29. Oktober 2012 .....