Design und Implementierung kryptographischer Funktionen und Protokolle des elektronischen Personalausweises

Clemens Deußer, Jan Reubold

30. November 2009

Eidesstattliche Erklärung des Prüfungsteilnehmers

Ich versichere durch meine Unterschrift an Eides statt, dass ich die Bachelorarbeit und die dazugehörige Dokumentation selbständig in der vorgesehen Zeit erarbeitet habe. Alle Stellen, die ich wörtlich oder annährend wörtlich aus Veröffentlichungen entnommen habe, wurden vom mir als solche kenntlich gemacht.

Name, Vorname: . . . . . .

Datum: . . . . . Unterschrift: . . . . . .

Name, Vorname: . . . . . .

Datum: . . . . . Unterschrift: . . . . . .

Diese Arbeit ist ein gemeinschaftliches Projekt von Jan Reubold und Clemens Deußer, sie wurde gemeinsam erstellt. Die abgegebene Version dieses Dokuments ist identisch für beide Abgaben. Wir haben zwar einige Abschnitte zusammen geschrieben, aber für die Bewertung wurde in jedem Kapitel ein entsprechender Verfasser notiert. Der Grund für die identische Abgabe ist, dass sich die Abschnitte aufeinander beziehen und zwei Dokumente mit ähnlichem Inhalt jedoch weniger Tiefe abzuschicken wäre nicht im Sinne eines ausführlichen Dokuments über das gesamte Projekt gewesen.

Falls dies Probleme bei der Bewertung verursacht bitten wir dies zu entschuldigen, auf diese Art ist ein besseres Dokument entstanden.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Der elektronische Personalausweis (ePA) ist zusammen mit der elektronischen Gesundheitskarte (eGK), der elektronischen Steuererklärung (ELSTER) und dem elektronischen Entgeltnachweis (ELENA) Teil der eCard Strategie der Bundesregierung. Die Zielsetzung dieser Strategie ist eine sichere, einfache und kostengünstige Authentifizierung, sowie eine Modernisierung und Effizienzsteigerung der öffentlichen Verwaltung, des Arbeits und Sozialwesens, sowie der Gesundheitsversorgung.

In diesem Dokument werden im allgemeinen diese eCards und im besonderen der elektronische Personalausweis (ePA) und die eID Applikation des ePA erörtert und eine Implementierung dieser Funktion beschrieben. Die eID Funktion dient der einfachen Authentifizierung durch zertifizierte Stellen. Dies können Behörden, aber auch private Unternehmen sein. Über die eID Funktion lassen sich Daten wie Name, Adresse und Geburtsdatum auslesen, welche nach wie vor auch auf dem Ausweis selbst zu sehen sind. Sie entspricht also einer authentischen elektronischen Version der visuellen Überprüfung eines solchen Ausweises.

Kapitel 2

Hintergrund

Dieses Kapitel wurde von Clemens Deußer verfasst.

2.1 Elektronische Identität

Eine Identität gehört immer zu einer Entität, also zu einem Individuum oder einer Gruppe die als Individuum auftritt, wie beispielsweise ein Unernehmen. Jedoch kann eine Entität mehr als eine Identität besitzen.

Vor allem im Internetzeitalter macht sich dies zunehmend deutlich, quasi virtuelle Identitäten werden ständig und beliebig erzeugt um eine Anonymität zu wahren. Zwar befindet sich hinter jeder Identität ein reales Individuum, aber abhängig von der Verwendung ist dies für den Betrachter irrelevant. Er kann den Zusammenhang zwischen verschiedenen Identitäten und dem dahinterstehenden Individuum oft nicht herstellen.

Um eine elektronische Identität also wirtschaftlich und sozial relevant zu machen, muss es sich um eine authentifizierte Identität handeln. Eine, welche den Zusammenhang zwischen Individuum und Identität so zweifelsfrei wie möglich wiedergibt. Es gibt zwar keine eindeutige Identität, aber eine realer Name und eine reale Adresse stellen den Zusammenhang im Rahmen eines staatlichen Gefüges mit einem minimalen Risiko verlässlich dar. Der ePA benutzt diese „reale“ Identität.

Die Identifikation ist dann der Prozess, in dem die angenommene Identität mit gespeicherten Daten abgeglichen wird. Die Authentifizierung ist der Prozess, der einer Identifikation ein genügend Maß an Vertrauen gibt, dass das Resultat der Identifikation korrekt ist.

Es ist wichtig zu erkennen, dass es keine absolute Sicherheit und kein absolutes Vertrauen gibt. Man kann diese Werte nur anhand eines Risikos und der Balance von miteinander konkurrierenden Interessen festlegen. Beispielsweise soll für ein hochvolumiges Geschäft ein höheres Maß an Vertrauen hergestellt werden als für privaten Schriftverkehr. Der private Schriftverkehr darf allerdings nicht nur an der wirtschaftlichen oder finanziellen Relevanz gemessen werden, da hier

datenschutzbedingte Interessen den Schutz der Identität erfordern. Der Begriff „genügend Maß an Vertrauen“ hängt also von vielen Variablen ab.

Man kann allerdings davon ausgehen, dass es keine obere Grenze für dieses Vertrauen gibt, sondern nur eine untere. Es kann nicht zu viel Vertrauen geben. Der ePA bemüht sich ein maximales Maß an Vertrauen herzustellen, dass die Identität auf dem ePA der Identität in dem Staat, also des realen Staatsbürgers, entspricht. Gleichzeitig verlangt der ePA auch die Authentifizierung der zugreifenden Instanz mit Zertifikaten. Das Vertrauen soll auf beiden Seiten hergestellt werden.

Ein prinzipielles Konzept macht die elektronische Identität angreifbar: Da der Ausweis seine Daten nicht nur visuell sondern auch digital preisgibt, besteht die prinzipielle Möglichkeit, diese Daten ohne Kontrolle des Inhabers auszulesen. Der ePA darf dies nicht zulassen, muss aber gleichzeitig bei Besitz des Ausweises freie Auslesbarkeit der Daten ermöglichen.

Als Beispiel: Der kontrollierende Polizist braucht nicht nachzufragen, wenn er den Ausweis in der Hand hält, aber er kann nicht in größerer Menge Profile erstellen indem er Ausweise ohne Wissen der Inhaber (z.B. aus der Ferne) ausliest.

Wie dies konkret umgesetzt wird, ist Teil des sicherheitsspezifischen Inhalt dieses Dokuments.

2.2 eCard Beschluss

Am 9. März 2005 beschloss das Bundeskabinett die Eckpunkte der eCard Strategie[2][13][12][1][8]:

- Die Bundesregierung will einen einheitlichen Standard schaffen, sowohl für eGovernment und eBusiness als auch für den privaten Gebrauch.
- Die eCards sollen Vorgänge in der Arbeits-, Sozial- und Gesundheitsverwaltung erheblich erleichtern, Dienste sollen kostengünstig, sicher und auf einem hohen Datenschutzniveau stattfinden.
- Alle Chipkarten sollen die Möglichkeit zur Qualifizierten Signatur (QS) beinhalten, dies entspricht einer elektronischen Unterschrift.
- Alle Verwaltungsvorgänge werden auf die Nutzung dieser Karten vorbereitet und Formerfordernisse werden überprüft und abgebaut. Auch mit dem Ziel der Entbürokratisierung.
- Herstellung der Karten und der Zertifikate (Trust Center) ist Aufgabe der Privatwirtschaft. [1][12][6]
- Die Karten sollen bestehende Karten nicht ersetzen, sondern sich in das Gesamtsystem einfügen.
- Man sollte anmerken, dass der elektronische Reisepass (ePass) nicht Teil dieser eCard Strategie ist. Er wurde bereits vorher beschlossen. Mehr dazu im nächsten Abschnitt.

[1] Nicht alle Zertifikate werden auf diese Weise vergeben, ein Teil der Zertifikate stammt von Ämtern (CSCA).

2.3 eCards im Überblick

Der elektronische Reisepass (ePass) wurde am 13. Dezember 2004 von dem Rat der EU verordnet und wird seit dem 1. November 2005 in Deutschland ausgegeben. Somit ist er nicht Teil des eCard Programms, sollte hier aber nicht unerwähnt bleiben.

Der ePA, die eGK, ELSTER und der Elena sind Teil der eCard Strategie. Sie sind ähnlich aufgebaut und verfolgen ein gemeinsames Ziel.

Quelle: [2][12][1]

• 2.3.1 ePass

Der ePass ersetzt den Reisepass für Auslandsreisen außerhalb der EU. Er beinhaltet - wie der Reisepass - personenbezogene Daten wie Adresse, Name und Gesichtsbild in elektronischer Form, sowie seit dem 1. November 2007 auch in Deutschland zwei Fingerabdrücke.

Das technische Konzept des ePass folgt den Empfehlungen der internationalen Zivilluftfahrtsorganisation ICAO (International Civil Aviation Organization), welches die technischen Standards für maschinenlesbare Reisedokumente festlegt.

Aufgrund der Resonanz gegenüber den biometrischen Merkmalen des ePass wurde bei der Einführung des ePA auf verpflichtende Verwendung dieser Merkmale verzichtet, die Möglichkeit wurde jedoch beibehalten.

• 2.3.2 eGK

Die elektronische Gesundheitskarte (eGK) enthält neben administrativen Daten elektronische Rezepte, Patientennotfalldaten und auch die Möglichkeit zur qualifizierten Signatur. Optional ist es so möglich, Verträglichkeiten von Medikamenten direkt auf der Karte zu überprüfen. Auch sollen Patientenüberweisungen erleichtert und kontrollierbarer werden. Bei über 700 Millionen Rezepten pro Jahr erhofft sich das Gesundheitsministerium Einsparungen von über einer Milliarde Euro.

• 2.3.3 ELSTER

Die elektronische Steuererklärung (ELSTER), soll, dem Namen entsprechend, die bisher papierbezogene Steuererklärung stark vereinfachen. Diese Funktion stellt zwar keine eigene Karte dar, ist aber mit den anderen Karten kompatibel, beispielsweise zu Zwecken der Authentifizierung. Zu diesem Zweck kann Benutzern dieser Funktion ein Zertifikat zur Verfügung gestellt werden.

- ### 2.3.4 ELENA

  Der elektronische Entgeltnachweis (ELENA), auch bekannt als JobCard, soll die Verwaltung von Arbeitsverhältnissen für sowohl Arbeitnehmer als auch Arbeitgeber erheblich vereinfachen. Im Rahmen der Einführung dieser Chipkarte wurden zusätzlich eine Reihe anderer Vorgänge der Arbeitsverwaltung modernisiert. So werden für Ansprüche wie Arbeitslosengeld und andere Sozialleistungen benötigte Daten nun zentral gespeichert.

- ### 2.3.5 ePA

  Der elektronische Personalausweis (ePA) soll den Personalausweis beginnend ab 1. Oktober 2010 schrittweise ersetzen. Er enthält alle personenbezogenen Daten, welche bisher auch auf dem Ausweis vorhanden waren in visueller und elektronischer Form, sowie eine elektronische Identifizierungsfunktion (eID) und die Möglichkeit biometrische Daten zu speichern. Zunächst freiwillig.

  Der ePA bildet den Fokus dieses Dokuments: Die Anwendungssoftware für diesen Ausweis wird aktuell entwickelt und dieses Projekt findet in diesem Kontext statt, somit wird dieser Ausweis Hauptgegenstand der folgenden Kapitel sein.

  Eine besondere Funktionalität des ePA ist die erwähnte eID Applikation. Mit dieser Funktion kann nicht nur eine Identität schnell überprüft werden, sondern auch eine einfache Altersüberprüfung stattfinden. Die Funktion ist insbesondere für die Verwendung in einer Online Umgebung vorgesehen, zur Authentifizierung wird entweder die CAN oder die PIN benötigt.

Alle Karten, Software und Zertifikate sollen von der Privatwirtschaft hergestellt und bereitgestellt werden, die Bundesregierung in Form des BSI definiert die Richtlinien und Standards.

Zusätzlich zu den elektronischen Merkmalen sollen die Ausweise einheitlich dem Scheckkartenformat ID-1 angepasst werden, im Fall des ePA entspricht dies einer deutlichen Verkleinerung des Ausweises.

2.4 Motivation des eCard Programms

Die Hauptmotivation hinter dem eCard Programm ist eine Anpassung der veralteten Ausweise an ein elektronisches Internet Zeitalter, es soll eine Vereinfachung und eine höhere Verlässlichkeit von Geschäften auf Internetbasis erreicht werden. Die Bundesregierung erhofft sich somit gesteigerte Wettbewerbsfähigkeit des Standortes Deutschland und einen Gewinn an Lebensqualität für die Einwohner, beispielsweise durch die Reduzierung von Behördengängen.

Viele Prozesse, welche den Ausweis bzw. eine Authentifizierung erfordern, können heute Online abgewickelt werden. Dafür sind die derzeitigen Ausweise

allerdings nicht geschaffen. Die Folge sind entweder eingeschränkte Verfügbarkeiten oder ein langwieriges Authentifizierungsverfahren durch das Schicken einer visuellen Kopie des Ausweises per Post.

Die visuellen Ausweise durch elektronische zu ersetzen soll also ein logischer Schritt sein, der sich nahtlos in eine moderne Gesellschaft einfügt.

Im Konkreten bedeutet dies eine Reihe von Vorteilen als Motivation[2]:

- Das eCard Programm soll den wirtschaftlichen Standort Deutschland stärken:
  - Durch vereinfachte Verwaltungsvorgänge im Arbeits- und Gesundheitswesen.
  - Durch erhöhten Konsum dank Erleichterung und Sicherung von Online Geschäften.
- Eine erhöhte Bürgernähe und Bürgerfreundlichkeit sichert die Akzeptanz dieses Projekts in der Bevölkerung.
- Die deutsche Verwaltung wird entbürokratisiert im Zuge dieser Umstellung.
- Durch die Bedrohung des internationalen Terrorismus werden Dokumente wie der ePass und auch der ePA immer wichtiger, um Attentätern die Einreise bzw. den Zugang zu erschweren.
- Fälschungssicherheit wird auf ein neues Niveau gehoben, Authentifizierungszeiten werden stark reduziert.
- In der administrativen Verwaltung und im Gesundheitssystem werden Kosten im Milliardenbereich gespart.

Modernisierung und Effizienzsteigerung sind allerdings nicht die einzigen Motivationen hinter dem ePA. Die Bundesregierung will generell eine höhere Authentizitätsrate, vor allem im Internet, durchsetzen, also einen höheren Anteil von authentifizierten Transaktionen, Geschäften und anderen Dienstleistungen, wie beispielsweise Foren. Das Internet ist ein „unsicheres“ Medium, in dem sich jeder Teilnehmer jederzeit in eine Anonymität flüchten kann. Dadurch werden illegale Aktivitäten leichter ermöglicht und ein generelles Misstrauen gegenüber Internetdienstleistungen wird gestärkt; dies soll geändert werden. Wenn man sich beispielsweise im Internet äußert oder ein Geschäft abschließt, so soll dies die gleichen Konsequenzen bzw. Möglichkeiten und Rechte beinhalten wie dies in einer „realen Situation“, also ausserhalb des Internets, der Fall wäre. Zwar bestehen die Regeln bereits und sie können durch die Aufnahme der IP-Adresse verfolgt werden, dies ist allerdings ein langwieriger und oft juristischer Prozess den sich viele kleine Firmen nicht leisten können und daher auf die Möglichkeit verzichten.

Dies könnte beispielsweise bedeuten, dass es Pflicht werden könnte Mitglieder einer Online Dienstleistung, wie etwa eines Forums, dazu zu verpflichten sich bei einer Registrierung auszuweisen. Somit werden Online Inhalte zuordnungsfähig und eventuelle Initiatoren von Mobbing, Nötigung und ähnlichem wären direkt greifbar und könnten zur Rechenschaft gezogen werden. Die datenschutzrechtlichen Implikationen eines solchen Schritts sind zu dem Zeitpunkt der Erstellung dieses Dokumentes noch nicht ausführlich öffentlich diskutiert worden.

2.5 Die eCard API

Die eCard API wurde im Zusammenhang mit der eCard Strategie beschlossen. Sie soll eine standardisierte Schnittstelle für alle eCards inklusive des ePass bereitstellen.

Im Gegensatz zur eCard API wird in dieser Arbeit *nicht* versucht eine allgemeingültige Schnittstelle zu schaffen, sondern nur eine Schnittstelle für den ePA und nur für die eID Funktion. Das allgemeine Design der eCard API, mit einer umfassenden Schnittstelle für alle Karten und alle Applikationen hat einige gravierende Nachteile.

Zum einen wird für die Installation durch die hohe Anzahl an verwendeten Bibliotheken bisher eine große Installationsdatei benötigt, während eine spezialisierte Applikation mit wesentlich weniger Platz auskommt. Zum anderen benutzt die eCard API große XML Header bei der Kommunikation, welche durch eine eventuelle Firewall interpretiert werden und die Transaktionen erheblich verlangsamen. So konnte es sein, dass bei der letzten Vorführung der eCard API die Ausführung eines eID Prozesses bis zu 15 Minuten dauerte, im Gegensatz zu den erstrebten 15 Sekunden.

Bei einer Abwägung von Adaptivität, Performance und Speicherverbrauch verzichtet man mit einer spezialisierten Applikation auf Adabptivität, gewinnt aber in den anderen Bereichen.

Die eID Funktion bildet voraussichtlich den Hauptbestandteil des ePA, sie ist für den Alltag ausgelegt und kann vor allem Online genutzt werden. Performance ist somit von größter Wichtigkeit. Die Nachfrage nach einer solchen, spezialisierten Identifizierungsfunktion ist beispielsweise im Onlinemarkt gegeben, falls ein Unternehmen nur diese Daten überprüfen will. Denkbare Anwendungen wären Bestellungen bei Versandhäusern oder Identifizierung als Mitglied bei diversen Vereinen und Foren.

Im Nachfolgenden wird die eCard API nach der Spezifikation des BSI, im Wortlaut des BSI definiert[6].

2.5.1 Application-Layer

Im Application-Layer befinden sich die verschiedenen Anwendungen, die das eCard-API-Framework für den Zugriff auf die eCards und damit verbundene Funktionen nutzen wollen. In dieser Schicht können auch weitere anwendungsspezifische „Convenience-Schnittstellen“ existieren, in denen wiederkeh-

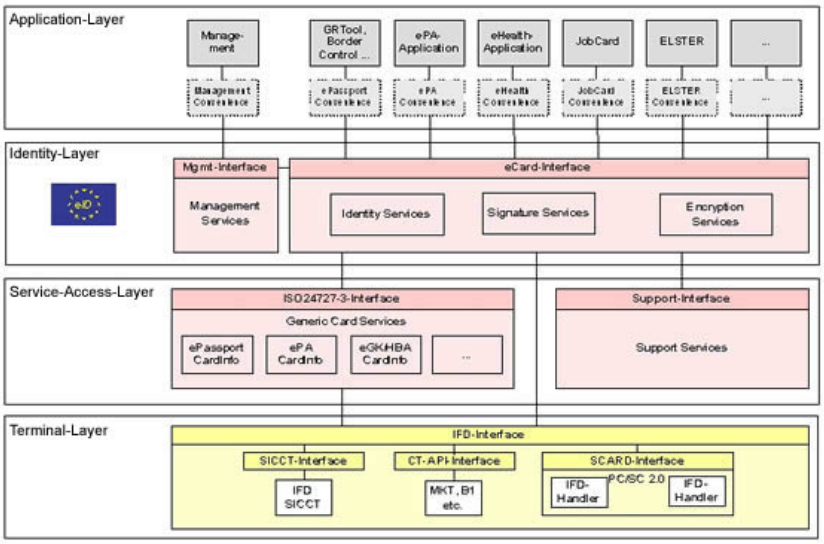

Abb. 2.1: Die verschiedenen Ebenen der eCard API; Quelle: BSI

rende Aufruffolgen in applikationsnahen Aufrufen gekapselt werden. Diese Schnittstellen sind jedoch derzeit nicht Gegenstand des eCard-API-Frameworks.

2.5.2 Identity-Layer

Der Identity-Layer umfasst das eCard-Interface und das Management-Interface und somit Funktionen für die Verwaltung und Nutzung elektronischer Identitäten sowie für das Management des eCard-API-Frameworks. Das eCard-Interface ermöglicht den Bezug von Zertifikaten sowie die Verschlüsselung, Signatur und Zeitstempelung von Dokumenten. Im Management-Interface existieren Funktionen für das Update des Frameworks und die Verwaltung von vertrauenswürdigen Identitäten, Chipkarten, Kartenterminals sowie des Default-Verhaltens.

2.5.3 Service-Access-Layer

Der Service-Access-Layer bietet insbesondere Funktionen für kryptographische Primitive und biometrische Mechanismen in Verbindung mit kryptographischen Token und umfasst das ISO24727-3-Interface und das Support-Interface. Das ISO24727-3-Interface ist eine Webservice-basierte Umsetzung des gleichnamigen Standards. Diese Schnittstelle enthält Funktionen, um (kryptographisch geschützte) Verbindungen zu Chipkarten herzustellen, Chipkartenapplikationen zu verwalten, Daten zu lesen oder zu schreiben, kryptographische Operationen auszuführen sowie das entsprechende Schlüsselmaterial (in Form von so genannten "Differential-Identities") zu verwalten. Hierbei sind alle Funktionen, die "Differential-Identities"nutzen oder verwalten, über protokollspezifische Object Identifier parametrisiert, so dass die unterschiedlichen in Teil 7 dieser Spezifikation definierten Protokolle über eine einheitliche Schnittstelle genutzt werden können. Das Support-Interface enthält eine Reihe von unterstützenden Funktionen.

2.5.4 Terminal-Layer

Der Terminal-Layer enthält im Wesentlichen das IFD-Interface. Dieses übernimmt die Generalisierung von konkreten Lesertypen und verschiedenen Schnittstellen sowie die Kommunikation mit der Chipkarte. Für den Anwender ist es nicht von Bedeutung, ob die eCard via PC/SC, einem SICCT-Leser oder einem herstellerspezifischen Interface angesprochen wird, ob sie kontaktbehaftet oder kontaktlos ist.

2.6 Der RFID Chip

Das Radio Frequency Identification RFID Prinzip (Identifizierung mit Hilfe von elektromagnetischen Wellen) bildet die Grundlage für die meisten per Funk auslesbaren Chips. Die Anwendungsbereiche dieser Chips sind nahezu unbegrenzt. So kann man Waren, Gegenstände, Autos für Mauterfassung, selbst Lebewesen und Ausweise mit diesen Chips ausstatten.

Die RFID Technologie besteht generell immer aus einem Chip als Transponder und einem Lesegerät. Der Chip selbst hat meistens keine eigene Stromversorgung, er wird von dem Lesegerät durch Induktionsspulen mit Strom versorgt. Dies funktioniert so, dass in dem Chip eine Antennenspule vorhanden ist, welche bei Kontakt mit einem elektromagnetischen Feld Induktionsstrom erzeugt. Der Chip speichert kleine Mengen dieses Stroms, decodiert die empfangenen Befehle und schickt die entsprechenden Antworten.

Da die Stärke eines elektromagnetischen Feldes quadratisch mit der Entfernung abnimmt haben die meisten RFID Chips nur eine kurze Reichweite. Falls ein RFID Chip jedoch über eigene Stromversorgung oder über unterstützende Stromversorgung verfügt, so kann die Reichweite beträchtlich erhöht werden. Zusätzlich zu der Stromversorgung ist auch die Art der eingesetzten Wellenfrequenzen für die Reichweite entscheidend. Mikrowellen erreichen die größte Reichweite, die Frequenz liegt zwischen etwa 300 MHz und 300 GHz.

Der Speicher auf einem RFID Chip variiert stark nach Einsatzgebiet und gewünschtem Gebrauch. Er kann von einem Bit bis zu mehreren Kilobytes enthalten, aber nach oben sind die Speicherkapazitäten nur durch Kosten und Platzbedarf begrenzt. Chips mit einem einzelnen Bit Kapazität werden beispielsweise in Warenhäusern zur Diebstahlprävention eingesetzt. Der Speicher ist generell beliebig oft auslesbar, aber meist nur einmal beschreibbar. Diese Chips besitzen beispielsweise eine statische Seriennummer und werden an oder in Gegenständen zur Identifikation angebracht.

Neuere Versionen dieser Chips, zu denen auch der ePA zählt, können ihre Nachrichten auch verschlüsseln und selber Berechnungen vornehmen.

2.7 Das MRTD Format

In diesem Abschnitt wird kurz ein Standardformat für Ausweise beschrieben, welches im weiteren Verlauf dieses Dokuments als Abkürzung benutzt wird.

MRTD steht für Machine Readable Travel Document (Maschinenlesbares Reisedokument). Es handelt sich um ein Format für Reisepässe und sonstige Ausweise. Alle derzeitig im Umlauf befindlichen Reisepässe, Personalausweise, alle eCards und der ePass entsprechen diesem Format. Die Bezeichnung „MRTD Chip“ bezeichnet also einen Chip auf einem MRTD kompatiblen Ausweis, mit MRTD kompatiblen Daten.

Das Format wurde von der Internationen Zivilluftfahrt-Organisation ICAO (International Civil Aviation Organization) entwickelt um einen Standard für die Überprüfung von Ausweisen zu schaffen. Zielsetzung war eine vereinfachter und sicherer Prozess für die Abfertigung von Ein- bzw. Ausreisenden, speziell an Flughäfen.

Kennzeichnend für diese Ausweise ist eine standardisierte Maschinenlesbare Zone auf Ausweisen, im allgemeinen zwei Zeilen am vorderen, unteren Rand des Ausweises. Dieses Konzept wurde später um elektronische Ausweise erweitert und alle eCards, insbesondere der ePA, sind zu diesem Format kompatibel. Auch biometrische Merkmale sind in dem MRTD Konzept geregelt. Während es früher oft nur ein Bild des Gesichts als einziges Merkmal gab, so wurde es später um Fingerabdrücke, Iris-Merkmale und der Möglichkeit weiterer Merkmale erweitert.

Wir verwenden die Bezeichnung „MRTD Chip“ ebenso wie die Bezeichnung „RFID Chip“. Während RFID ein technologisches Prinzip für Chiparchitektur ist, handelt es sich bei MRTD um ein standardisiertes Format. Es bestimmt Anforderungen, denen ein Ausweisdokument genügen muss um kompatibel zu dem MRTD Framework zu sein. Ein RFID Chip ist somit ein per Funk auslesbarer Chip, ein MRTD Chip ist ein Reisedokument was zur Authentifizierung zum Beispiel an Flughäfen verwendet werden kann.

Kapitel 3

Diffie Hellman und Elliptische Kurven

In diesem Kapitel werden einige mathematische und kryptographische Konstrukte, Funktionen oder Protokolle beschrieben. Die Intention ist eine Übersicht über diese zu geben und außerdem die Verständlichkeit für Personen, welche zwar kryptographisch kompetent, aber nicht mit den Protokollen oder dem Kontext vertraut sind, zu gewährleisten.

Sie sind in mathematischem oder kryptographischem Sinne zwar nicht falsch, aber nicht komplett umfassend und nicht absolut exakt beschrieben, da dies zwangsläufig entweder mit dem Umfang oder der Verständlichkeit des Kapitels kollidieren würde.

Dieses Kapitel wurde von Clemens Deußer verfasst.

3.1 Diffie-Hellman

Das Diffie-Hellman Protokoll wurde von Martin Hellman, Whitfield Diffie und Ralph Merkle entwickelt. Es beschreibt ein Protokoll, welches anhand einer unsicheren Kommunikation trotzdem einen sicheren Kommunikationskanal ermöglicht.

Der Ablauf stellt sich wie folgt dar; die beiden Parteien, sagen wir Alice und Bob, verständigen sich erst auf gemeinsame Parameter: Eine Basis g und einen primen Modulus p. Der Modulus wird auf alle berechneten Zahlen in dem Protokoll angewandt und findet keine weitere Erwähnung in dem folgenden Text.

Zunächst erzeugen Alice und Bob jeweils einen Parameter $a, b \in \{1, \ldots, p-1\}$, berechnen $A = g^a$ bzw $B = g^b$ und schicken diese Zahl an die jeweils andere Partei, so dass nun Alice und Bob zusätzlich zu den zu Beginn vereinbarten Parametern g und p im Besitz von $\{a, B\}$ bzw $\{b, A\}$ sind. Ein lauschender Beobachter hingegen kennt nur $\{g, p, A, B\}$.

Sowohl Alice als auch Bob berechnen nun A^b bzw B^a, diese beiden Zahlen sind identisch, es gilt:

$$A^b = (g^a)^b = g^{ab} = (g^b)^a = B^a$$

Der lauschende Beobachter kann diese Zahl nicht berechnen, da er dafür den diskreten Logarithmus von A oder B berechnen müsste, dies aber entspricht einer schweren Operation, zu deren Berechnung derzeit keine effiziente Möglichkeit bekannt ist. Falls man die Zahlen groß genug wählt ist ein Brute Force Ansatz in einem sinnvollen zeitlichen Rahmen nicht mehr durchführbar, ohne die Berechnung der Schlüssel zu Zeitaufwändig zu gestalten.

Das Ergebnis wird nun benutzt um eine sichere Kommunikation aufzubauen.

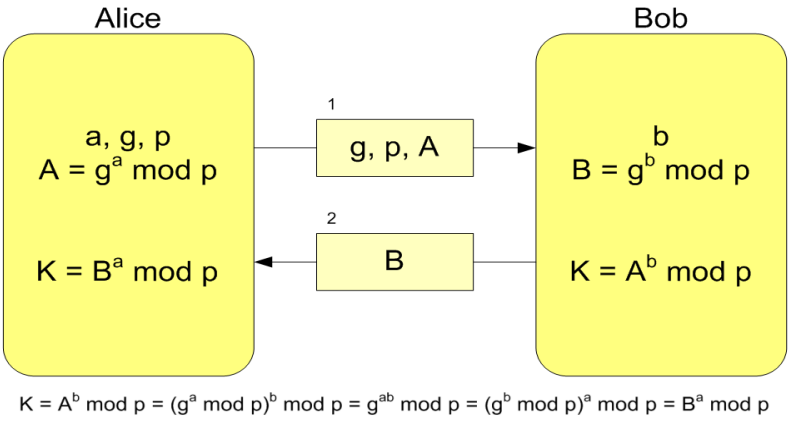

Abb. 3.1: *Darstellung eines Diffie-Hellman Schlüsselaustauschs; Quelle: Wiki Commons*

Diffie-Hellman kann durch eine sogenannte „Man in the middle“ Attacke angegriffen werden. Hierbei sitzt der Angreifer zwischen den beiden Parteien, fängt die Nachrichten auf beiden Seiten ab und führt faktisch zwei Diffie-Hellman Protokolle (eins mit jedem Teilnehmer) unbemerkt von den Teilnehmern durch. Bei der nachfolgenden Kommunikation entschlüsselt er eingehende Pakete, verschlüsselt sie wieder und schickt sie weiter. Dadurch kann er die Daten lesen und sogar verändern, ohne dass dies bemerkt wird.

Möglichen Schutz gegen diesen Angriff bietet beispielsweise eine Signatur, im Falle von PACE werden neue Parameter anhand eines Passworts berechnet, somit kann der Angreifer die Berechnungen nicht mehr korrekt durchführen (siehe PACE).

3.2 Elliptische Kurven

Elliptische Kurven stehen nicht im Fokus dieser Arbeit, da wir allerdings ECDH (Elliptic Curve Diffie-Hellman) benutzen sollte das Prinzip dahinter nicht unerwähnt bleiben.

Abstrakt ist eine elliptische Kurve eine singularitätenfreie algebraische Kurve der dritten Ordnung in der projektiven Ebene. Also eine „glatte“ Kurve ohne Singularitäten, welche durch eine Polynomgleichung dritten Grades beschrieben werden kann und in der projektiven Ebene liegt.

Für die Kryptographie sind einige wichtige Eigenschaften dieser Kurven von Bedeutung. Im Besonderen gilt die Multiplikation auf dieser Kurve als „schwere“

Operation und die Schlüssellänge ist bedeutend geringer im Vergleich zu einer RSA Verschlüsselung.

Alle elliptischen Kurven lassen sich durch die Formel $y^2 = x^3 + ax + b$ darstellen, abhängig von den Variablen a und b lassen sich so unterschiedliche Kurven in einem bestimmten Bereich erstellen. Für kryptographische Zwecke ist es sinnvoll nur bestimmte vordefinierte elliptische Kurven zu verwenden, da die Stärke der Verschlüsselung von der Wahl der Kurve abhängt und die Feststellung der Eignung einer solchen Kurve nicht trivial ist. Institute wie beispielsweise das NIST (National Institute of Standards and Technology) stellen solche Kurven bereit.

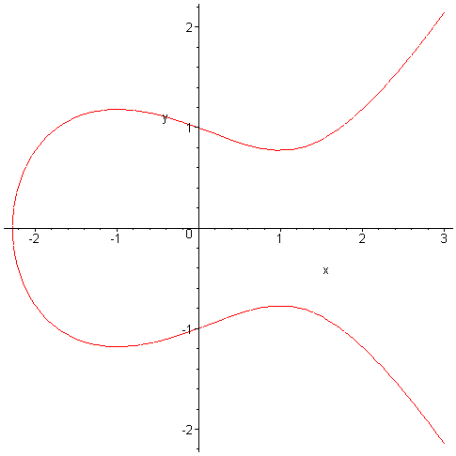

Abb. 3.2: *Eine typische elliptische Kurve; Quelle: Wiki Commons*

Punkte auf elliptischen Kurven betrachten wir als Elemente einer abelschen Gruppe, also eine Grupper G innerhalb derer das Kommutativgesetz für alle Elemente der Gruppe gilt, ein Beispiel für die Addition: $a+b = b+a$ für alle $a, b \in G$.

Die Addition auf elliptischen Kurven ist wie folgt definiert:

Der Punkt im Unendlichen ist als 0 definiert. Die Summe von drei Schnittpunkten einer Geraden mit der Kurve ergibt 0, somit ist die Summe zweier Punkte das Inverse des dritten Schnittpunkts der Verbindungsgeraden. Falls eine Gerade nur zwei Schnittpunkte besitzt, so ist die Summe dieser beiden Punkte definiert als 0. Die Summe eines Punktes mit sich selbst (Verdopplung) ist das inverse des zweiten Schnittpunkts der Tangenten durch diesen Punkt. Tangenten schneiden in dem tangierten Punkt die Gerade zweimal, somit ist dies konsistent mit der oben genannten Definition.

Die skalare Multiplikation gehorcht den Regeln der Addition, so kann man beispielsweise einen Punkt verdoppeln und den entstandenen Punkt wieder verdoppeln, was zwei Multiplikationen mit 2, also einer Multiplikation 4 entsprechen würde. Dementsprechend kann jede Multiplikation durch eine Serie von Verdopplungen und Additionen durchgeführt werden.

Diese elliptischen Kurven finden verstärkt Anwendung bei Public Key Kryptographie. Für diese Art der Kryptographie benötigt man mathematische Vor-

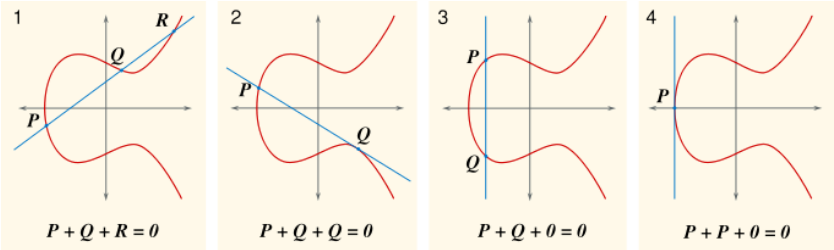

Abb. 3.3: Diverse Fälle der Addition auf elliptischen Kurven; Quelle: Wiki Commons

gänge welche sehr effizient in eine Richtung verfolgt werden können, jedoch schwer in die andere.

RSA Public Key Verschlüsselung, stark simplifiziert, stützt sich auf die Schwierigkeit, das Produkt zweier hoher Primzahlen zu faktorisieren. Es existiert kein effizienter Algorithmus und die Zahlen sind groß genug um einen Brute Force Ansatz in einem zeitlich sinnvollen Rahmen zu verhindern, jedoch wurde niemals bewiesen, dass ein solcher Algorithmus nicht existieren kann.

Wie bei RSA muss bei elliptischen Kurven ein Faktor gefunden werden, allerdings in diesem Falle der skalare Faktor einer Multiplikation mit einem Punkt.
Skalare Multiplikation eines Punktes ist wie oben erwähnt sehr effizient mit einer Serie von Verdopplungen und Additionen durchzuführen (hierbei existieren einige Algorithmen zur zusätzlichen Optimierung). Aus dem Produkt jedoch den skalaren Faktor zu bestimmen ist schwer. Der Brute Force Ansatz, einfach hochzuzählen, ist bereits für relativ kleine Schlüsselgrößen nicht praktikabel und es existiert noch kein effizienter Algorithmus für dieses Problem. Wie jedoch bei allen anderen Public Key Kryptosystemen ist auch hier die Existenz eines solchen Algorithmus *nicht* widerlegt und die Schlüssellänge muss mit steigender Computerkapazität und dem Finden effizienterer Brute Force Ansätze erhöht werden.

Ein Beispiel:
Man kann einen Punkt P mit der Zahl 68719476736, was (praktischerweise) 2^{36} entspricht, multiplizieren indem man den Punkt 36 mal verdoppelt, also 36 Additionen durchführt, sehr effizient und einfach. Um den Faktor aus dem Produkt zu bestimmten müsste man allerdings aufgrund fehlender Möglichkeiten zur Abschätzung den Punkt P schrittweise addieren und selbst mit effizienteren Brute Force Ansätzen wäre man bei den benötigten Additionen noch im Bereich der Größenordnung des Faktors, also deutlich mehr als bei der Berechnung des Produkts.

3.3 Elliptic Curve Diffie-Hellman

In dieser Variation des Diffie-Hellman Schlüsselaustauschs wird die Sicherheit in elliptischen Kurven gewährleistet, daher sind auch die zugrunde liegenden Parameter andere als in der Standardversion.

Alle Parteien müssen sich zunächst auf alle Parameter der zugrunde liegenden elliptischen Kurve einigen, dazu werden nicht nur die Parameter a und b der elliptischen Kurve benötigt, sondern auch ein primer Field Identifier p, sowie der Erzeuger G einer zyklischen Untergruppe $E(\mathbb{F}_p)$, dessen Ordnung n (kleinste Zahl, so dass gilt $nG = 0$) und der Kofaktor $h = \frac{|E|}{n}$. Der Kofaktor ist eine natürliche Zahl und es muss gelten $h < 4$ (in unserem Fall gilt immer $h = 1$).

Die Parameter sind somit $D = \{p, a, b, G, n, h\}$.

Wobei G ein Punkt auf der Kurve ist und alle anderen Parameter natürliche Zahlen sind. Die Sicherheit der Kurve wird durch die Größe des Field Identifiers p definiert; die bis heute höchste geknackte Schlüssellänge einer elliptischen Kurve beträgt 112 bit, für reale Verschlüsselungen werden üblicherweise Bitlängen von 160 bis 384 bits verwendet.

Der weitere Schlüsselaustausch findet statt wie in dem ursprünglichen Diffie-Hellman Protokoll:

Beide Parteien bestimmen eine Zufallszahl $x, y \in \{1, \ldots, p-1\}$ (statt a, b) und berechnen $X = xG$ bzw $Y = yG$ (statt $A = g^a$ bzw $B = g^b$). Diese Ergebnisse werden verschickt und beide Parteien berechnen das Geheimnis $P = xY = xyG = yX$ (entsprechend zu $K = A^b = g^{ab} = B^a$).

Mit der Berechnung von P ist der Durchlauf des Protokolls abgeschlossen.

Wir verwenden in dieser Arbeit ausschließlich den elliptic curve Modus des Diffie-Hellman Schlüsselaustauschprotokolls.

Kapitel 4

Hilfsfunktionen und Formate

Die erste Hälfte dieses Kapitels bis einschließlich Key Derivation Function wurde von Clemens Deußer verfasst, die folgenden Abschnitte von Jan Reubold.

4.1 APDU Kommunikation

Die gesamte Kommunikation zwischen Chip und Terminal erfolgt durch sogennante APDUs (Application Protocol Data Units)[5].

APDUs unterteilen sich in CAPDUs (Command APDUs) und RAPDUs (Response APDUs), CAPDUs entsprechen einer Anfrage und RAPDUs stellen die entsprechende Antwort auf eine CAPDU dar.

Eine CAPDU besteht aus einem Klassenbyte, einem Instructionbyte, zwei Befehlsbytes, einem Byte für die Länge der beinhalteten Daten, die entsprechenden Daten und ein optionales Längenbyte für die erwartete Länge der Antwortdaten.

Die Datenfelder von unerweiterten APDUs können maximal 255 bytes für CAPDUs und 256 bytes für RAPDUs enthalten. Dies begründet sich darin, dass der Längenparameter der CAPDU nur Werte zwischen $1 \ldots 255$ annehmen kann, während die erwartete Länge auch auf $0x00 \widehat{=} 256$ gesetzt werden kann. Eine erweiterte APDU kann bis zu 65536 bytes enthalten, da sie zwei Längenbytes zur Verfügung stellt, ist also für die Anwendung bei Chipkarten praktisch nicht limitiert. Um eine erweiterte APDU zu verwenden muss man anstatt des Längenbytes Lc bzw. Le drei Bytes verwenden, wobei das erste Byte als 0 gesetzt wird und die folgenden zwei Bytes der Länge entsprechen.

Der Header muss immer angegeben werden, die Angabe von Daten und der erwarteten Länge ist optional.

Dementsprechend gibt es vier Möglichkeiten für die Zusammensetzung:

Eine RAPDU besteht, falls vorhanden, aus den entsprechenden Antwortdaten, begrenzt durch den Längenparameter in der vorangegangenen CAPDU, und zwei Statusbytes.

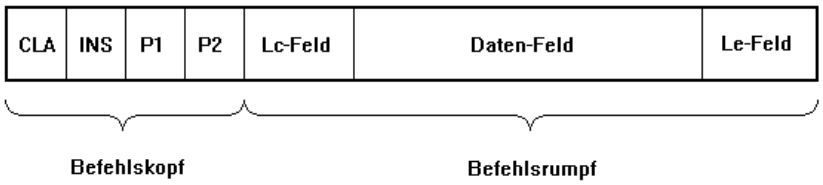

Abb. 4.1: Anordnung der Felder in einer CAPDU; Quelle: TU-Berlin

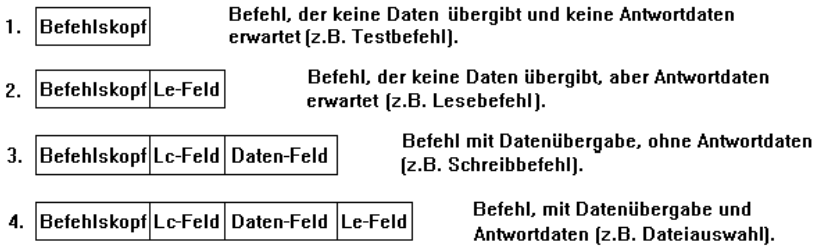

Abb. 4.2: Möglichkeiten der Zusammensetzung einer CAPDU; Quelle: TU-Berlin

Falls die Operation ohne Fehler beendet wurde, so werden die Statusbytes nach den Antwortdaten angezeigt und zeigen im Allgemeinen 90 00 oder 61xx an (kein Fehler).

Falls während des Vorgangs ein Fehler auftrat, so werden grundsätzlich keine Antwortdaten geschickt und die Statusbytes zeigen den entsprechenden Fehler anhand eines Codes an.

Damit man innerhalb eines Datenfeldes (sowohl CAPDU als auch RAPDU) mehr als eine einzelne Information übermitteln kann, werden sogenannte Tags verwendet. Diesen Tags folgt ein Längenparameter, der die Länge der folgenden Information angibt. Dies kann man beliebig schachteln, also kann man auch Tags mit Informationen in andere Informationen einfügen oder ganze Objekte erzeugen, welche aus verschiedenen Tags und deren Informationen bestehen.

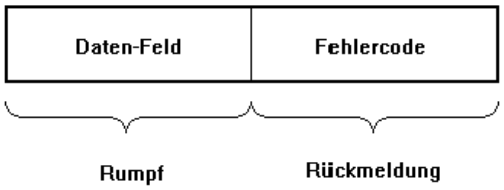

Abb. 4.3: Anordnung einer RAPDU; Quelle: TU-Berlin

4.2 ASN.1

ASN.1 ist eine allgemeine Sprache zur abstrakten Definition und Übertragung von bestimmten Vorgängen, Datenstrukturen und allgemeinen Standards. Eine in ASN.1 verfasste Information ist keine Implementierung, sondern eine reine Definition.

ASN.1 wurde usprünglich von ISO/IEC (International Organisation for Standardization) und ITU-T (Telecommunication Standardization Sector) im Jahre 1984 entwickelt und wurde 1988 mit der Veröffentlichung der Version X.208 eine selbstständige Sprache für Standards.

ASN.1 wird in fast allen Bereichen wo Standards gebraucht werden oder mathematische Strukturen für Programme benötigt werden eingesetzt. In diesem Dokument findet diese Sprache Einsatz bei der Kodierung von diversen Parameterumgebungen auf dem MRTD Chip[5][15].

4.2.1 Encoding Rules

Wenn ASN.1 die Programmiersprache ist, so sind die Encoding Rules (Regeln zur Verschlüsselung) die „Compiler“. Sie definieren die Art der Kodierung bei Übertragungen, so dass eine ASN.1 Struktur als Bitstring verschickt werden kann und mit dem gleichen Satz von Regeln wieder als ASN.1 Struktur darstellbar ist.

Die wohl bekannteste Encoding Rule ist die „Distinguished Encoding Rule“ (DER), diese wird auch von dem MRTD Chip benutzt.

DER benutzt einfache Tags und Längenbytes zur Kodierung der ASN.1 Struktur, ähnliche Regeln wie auch in der APDU Struktur verwendet.

4.2.2 Object Identifier

Object Identifiers stehen nicht direkt in Verbindung zu ASN.1, werden aber oft im Kontext von ASN.1 Strukturen benutzt.

Ein Object identifier (OID) ist ein weltweit eindeutiger Bezeichner für beliebige Datenobjekte und Informationen. Er ist ähnlich aufgebaut wie eine Internet IP Adresse, unterteilt in sogenannte „Knoten“, dargestellt durch Punkte in dem identifier. Ein Knoten steht für eine bestimmte Instanz, welche darunterliegende Knoten an andere Instanzen vergibt, bis dann schlieSSlich die Verbindung der Knoten mit den letzten Identifikatoren das Datenobjekt eindeutig definieren.

In diesem Dokument werden OIDs hauptsächlich eingesetzt um benutzte Protokolle zu definieren, so steht zum Beispiel der OID „0.4.0.127.0.7.2.2.4.2.1“ für BSI (0.4.0.127.0.7), BSI-de Protokolle (2), Smartcard (2), PACE (4), ECDH (2), 3DES (1) (Quelle: Technische Richtlinien des BSI).

Die gesamte OID identifiziert also das PACE Protokoll in Elliptic Curve Diffie-Hellman Modus mit Triple DES Verschlüsselung des Bundesministeriums für Sicherheit in der Informationstechnik. Dieser bestimmte OID bedeutet nur dies und kann keine weiteren Bedeutungen haben, da er weltweit eindeutig ist.

4.3 Java Cryptographic Architecture

Die Java Cryptographic Architecture (JCA) ist ein Sicherheits Framework von Java, es wird unter anderem von dem Flexiprovider Dienst benutzt, welchen wir in unserer Arbeit sehr stark verwenden.

Das Framework wird benutzt für die Implementierung von Algorithmen zu kryptographischen Zwecken, das Design Prinzip lässt sich in den folgenden Punkten zusammenfassen:

- Implementierungsunabhängigkeit und Interoperabilität und
- Algorithmenunabhängigkeit und Erweiterbarkeit.

Implementierungsunabhängigkeit bedeutet, dass verschiedene Implementierungen des gleichen Algorithmus benutzt werden können, ohne einen Unterschied in der Verwendung zu bewirken. Falls eine bestimmte Implementierung gewünscht wird, so kann diese spezifisch angegeben werden. Gleichzeitig können auch Implementierungen zusammenarbeiten, ohne dafür speziell ausgelegt zu sein, sie sind interoperabel.

Vollständige Unabhängigkeit von Algorithmen lässt sich nicht erreichen aufgrund der stark unterschiedlichen Charaktere verschiedener Algorithmen. Die JCA stellt in diesem Fall ein möglichst umfangreiche Menge an Klassen und standardisierten APIs (Application Programming Interface, Programmierschnittstelle) bereit, um die Verwendung von Algorithmen in einer bestimmten Implementierung zu ermöglichen ohne tiefgehendes Wissen des Algorithmus zu besitzen. Diese Klassen werden „Engine Klassen“ genannt.

Erweiterbarkeit von Algorithmen bedeutet das leichte Hinzufügen von Algorithmen, die einer dieser „Engine Klassen“ angehören.

4.3.1 Design

Hauptsächlich besteht die JCA aus zwei Komponenten: Dem Framework, welches die kryptographischen Dienste definiert und bereitstellt, und die Provider, welche diese implementieren.

Ein Provider, im folgenden CSP genannt (Cryptographic Service Provider), stellt also die Implementierung eines oder mehrerer Algorithmen dar, er muss alle implementierten Dienste und Algorithmen auflisten und falls eine bestimmte Implementierung gesucht wird und in dieser Liste enthalten ist, so wird eine Instanz dieses Providers erstellt und benutzt. Die eigentliche Implementierung wird von dem Provider in bestimmten Packages bereitgestellt, welche in Zusammenhang mit der Liste verbunden werden. Der Programmierer hat somit genügend Spielraum, um den Algorithmus schnittstellengerecht möglichst ungehindert durch das Framework zu implementieren.

Um einen Provider zu benutzen, ruft man beispielsweise innerhalb eines Programms die entsprechende „Engine Klasse“ auf und fordert die gewünschte Instanz, optional mit dem konkreten gewünschten Provider.

Innerhalb der JCA existieren bestimmte Prioritäten unter Providern, so können zwei verschiedene Provider das selbe implementieren, jedoch beim Abruf

dieser Implementierung wird immer der höher priorisierte benutzt. Die einzige Möglichkeit dies zu umgehen ist die Angabe eines konkreten Providers.

Provider sind im allgemeinen vordefiniert und oft bereits in vielen Standard Java Bibliotheken enthalten, aber dieses Design ermöglicht und erleichtert die eigene Implementierung von Providern und deren Benutzung, wie in unserem Falle des Flexiproviders der TU-Darmstadt, der Endnutzer kann sehr einfach neue Provider hinzufügen und sofort verwenden.

4.3.2 Beispiele

Um das JCA Framework zu benutzen, reicht es sich eine Instanz aus einer der „Engine Klassen“ zu holen.

So könnte man beispielsweise für die Verwendung des MD5 Hash folgendes Code Fragment benutzen:

```
MessageDigest md = MessageDigest.getInstance("MD5");
```

Mit Aufruf der „getInstance“ Methode fragt die Engine Klasse „MessageDigest“ das JCA Framework nach dem ersten Provider welcher den genannten Algorithmus, in diesem Fall MD5, unterstützt. Falls man einen bestimmten Provider wünscht, so muss man diesen in einem zweiten Parameter angeben, beispielsweise:

```
MessageDigest md = MessageDigest.getInstance("MD5", "FlexiCore");
```

In diesem Fall wird nach dem FlexiCore Provider gesucht und, falls er den genannten ALgorithmus unterstützt, zurückgegeben.

4.4 Key Derivation Function

Eine Key Derivation Function (Schlüsselherleitungsfunktion, KDF) dient generell zum Herleiten eines Schlüssels aus einer beliebigen, geheimen Information in Form einer Zahl.

In unserem Fall wird eine solche KDF für drei Fälle benutzt: Für Secure Messaging, für das Erstellen eines MAC Schlüssels und für das Herleiten eines Schlüssels aus einem Passwort.

Diese KDF hat generell die Struktur $\mathbf{KDF}(K, [r], c)$ wobei K das Geheimnis ist, r eine optionale Nonce (Number used Once, eine Zufallszahl) und c ein 32-bit Integer abhängig von dem verwendeten Modus.

Für den Secure Messaging Modus gilt: $\mathbf{KDF}_{Enc}(K, [r]) = \mathbf{KDF}(K, [r], 1)$

Für den MAC Schlüssel gilt: $\mathbf{KDF}_{MAC}(K, [r]) = \mathbf{KDF}(K, [r], 2)$

Für den Passwortschlüssel gilt: $\mathbf{KDF}_{\pi}(\pi) = \mathbf{KDF}(f(\pi), 3)$, wobei π das Passwort ist und f(π) eine Character String Funktion nach ISO/IEC 8859-1 (siehe Abb.).

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F
1-	0010	0011	0012	0013	0014	0015	0016	0017	0018	0019	001A	001B	001C	001D	001E	001F
2-	0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3-	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4-	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5-	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
6-	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7-	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	007F
8-	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	008A	008B	008C	008D	008E	008F
9-	0090	0091	0092	0093	0094	0095	0096	0097	0098	0099	009A	009B	009C	009D	009E	009F
A-	00A0	Ą 0104	˘ 02D8	Ł 0141	¤ 00A4	Ľ 013D	Ś 015A	§ 00A7	¨ 00A8	Š 0160	Ş 015E	Ť 0164	Ź 0179	- 00AD	Ž 017D	Ż 017B
B-	° 00B0	ą 0105	˛ 02DB	ł 0142	´ 00B4	ľ 013E	ś 015B	ˇ 02C7	¸ 00B8	š 0161	ş 015F	ť 0165	ź 017A	˝ 02DD	ž 017E	ż 017C
C-	Ŕ 0154	Á 00C1	Â 00C2	Ă 0102	Ä 00C4	Ĺ 0139	Ć 0106	Ç 00C7	Č 010C	É 00C9	Ę 0118	Ë 00CB	Ě 011A	Í 00CD	Î 00CE	Ď 010E
D-	Đ 0110	Ń 0143	Ň 0147	Ó 00D3	Ô 00D4	Ő 0150	Ö 00D6	× 00D7	Ř 0158	Ů 016E	Ú 00DA	Ű 0170	Ü 00DC	Ý 00DD	Ţ 0162	ß 00DF
E-	ŕ 0155	á 00E1	â 00E2	ă 0103	ä 00E4	ĺ 013A	ć 0107	ç 00E7	č 010D	é 00E9	ę 0119	ë 00EB	ě 011B	í 00ED	î 00EE	ď 010F
F-	đ 0111	ń 0144	ň 0148	ó 00F3	ô 00F4	ő 0151	ö 00F6	÷ 00F7	ř 0159	ů 016F	ú 00FA	ű 0171	ü 00FC	ý 00FD	ţ 0163	˙ 02D9

Abb. 4.4: ISO/IEC 8859-1 Char Set; Quelle: UC Berkeley School of Information

Die KDF selbst berechnet lediglich den Hashwert der konkatenierten angegebenen Parameter $h = \mathbf{H}(K \parallel r \parallel c)$ und berechnet daraus eine Octet String Ausgabe.

Quelle: BSI Technische Richtlinien[5].

4.5 Auslesen der EF.CardAccess Datei

In diesem Abschnitt wird auf die EF.CardAccess Datei, genauer auf das Auslesen dieser, sowie auf das Selektieren bestimmter, in der Datei enthaltener, Informationen eingegangen.

Zum Auslesen dieser Datei ist keine Authentifizierung erforderlich, sie steht zur freien Verfügung.

Das Schreiben, also das Editieren der Informationen, auf der Datei ist hingegen unmöglich, es gibt kein Protokoll oder eine andere Möglichkeit Schreibrechte auf diese Datei zu erhalten.

Die, auf der EF.CardAccess Datei enthaltenen Informationen liegen als ASN1 Datenstruktur SecurityInfo in DER-Codierung vor. Mit dem entsprechenden OID können somit die gewünschten Daten ausgelesen werden.

```
SecurityInfos ::= SET OF SecurityInfo

SecurityInfo ::= SEQUENCE
  protocol OBJECT IDENTIFIER,
  requiredData ANY DEFINED BY protocol,
  optionalData ANY DEFINED BY protocol OPTIONAL
```

Diese Datei enthält alle relevanten SecurityInfos (siehe unten), um die Applikationen ausführen zu können:

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

Sollte man beispielsweise die Domain Parameter für PACE mit Elliptic Curve Diffie-Hellman auslesen wollen, müsste man zunächst mit einem Select-Befehl die Master File auswählen und dann den entsprechende Unterordner.

Sobald man die Datei mit einem Select-Befehl ausgewählt hat, kann man beginnen die CardAccess Datei auszulesen.

Da maximal 256 Bytes auf einmal geschickt werden können, die Datei jedoch im Allgemeinen größer ist, werden die benötigten Daten schrittweise ausgelesen.

Sobald man die gesamte Datei ausgelesen hat, werden die Informationen in ein Array von SecurityInfos gesteckt.

Nun kann man, mit bestimmten OIDs, die gewünschten Informationen auslesen.

4.6 Auslesen der EF.CardSecurity Datei

In diesem Abschnitt wird auf die EF.CardSecurity Datei, genauer auf das Auslesen dieser, sowie auf das selektieren bestimmter, in der Datei enthaltener, Informationen eingegangen.

Zum Auslesen dieser Datei ist, im Gegensatz zur EF.CardAccess Datei, das vorherige Ausführen von PACE und TA notwendig.

Das Schreiben, also editieren der Informationen ist hingegen, wie bei der EF.CardAccess Datei unmöglich.

Die, auf der EF.CardSecurity Datei enthaltenen Informationen liegen als ASN1 Datenstruktur SignedData in DER-Codierung vor.

SignedData ::= SEQUENCE

version CMSVersion,
digestAlgorithms DigestAlgoritmsIdentifiers,
encapContentInfo EncapsulatedContentInfo,
certificates [0] IMPLICIT CertificateSet OPTIONAL,
crls [1] IMPLICIT RevocationInfoChoices OPTIONAL
signerInfos SignerInfos

Quelle: [10]

Diese Datei enthält alle vorhandenen signierten SecurityInfos.

Das Auslesen der Datei erfolgt analog zu dem Auslesen der EF.CardAccess Datei, nur das jegliche Nachrichten als Secure Message verschickt werden.

Sobald man die gesamte Datei ausgelesen hat, werden die Informationen in ein SignedData Objekt gesteckt.

Nun kann man die gewünschten Informationen auslesen.

4.7 Die PKI des MRTD Chips

Der MRTD Chip bietet eine PKI an, da sich beispielsweise das Terminal für bestimmte Funktionalitäten bei der Karte mit Zertifikaten ausweisen muss.

Das Zertifikat muss in diesem Fall mindestens ein Terminal Zertifikat besitzen, welches einen öffentlichen Schlüssel, sowie die Zugriffsrechte enthält. Außerdem muss der Gültigkeitszeitraum des Zertifikates angegeben sein. Es muss ein Feld mit dem Datum des Inkrafttretens des Zertifikates, sprich dessen Erstellungsdatum, und ein Feld mit dem Verfallsdatum des Zertifikates beinhalten.

Da der MRTD Chip eine relativ geringe Rechenleistung hat, müssen die Zertifikate als Card Verifiable Certs (CVCs) vorliegen. Dies gilt für CVCA Link Zertifikate genauso, wie für DV- und Terminal Zertifikate.

Es gibt in dieser Infrastruktur drei verschiedene Instanzen:

- Country Verifying CAs (CVCAs)
- Document Verifier (DVs)
- Terminals

Für genauere Informationen siehe unten.

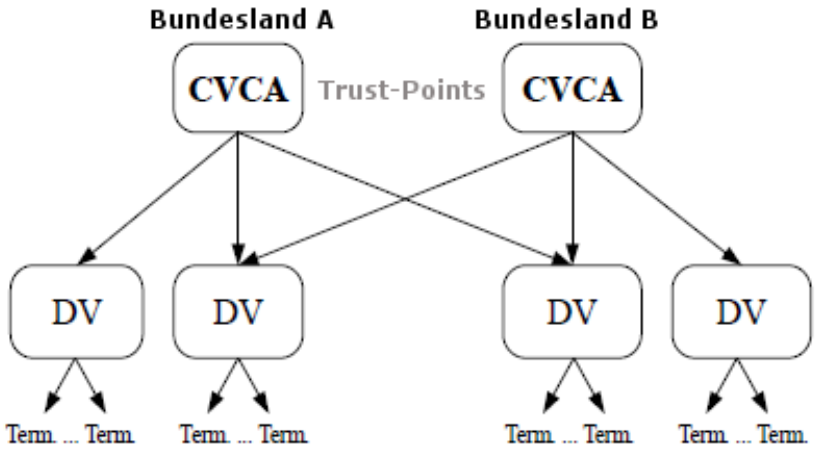

Abb. 4.5: PublicKey Infrastruktur des MRTD Chips

Ein MRTD Chip besitzt mindestens einen und maximal zwei sogenannte "Trust-Points" (CVCAs), diese werden in der Produktionsphase auf dem Chip gespeichert.

Sobald die Karte eine Zertifikatskette erhält, verifiziert sie als erstes anhand der "Trust-Points" und der Signatur, ob das CVCA Zertifikat gültig ist. Sollte dies der Fall sein, überprüft sie, ob jedes darauffolgende Zertifikat in der Kette von dem vorhergehenden erstellt wurde, und ob es noch gültig ist.

Hierbei gibt es zwei wichtige Werte auf Seiten der Karte, die zu beachten sind.

Erstens muss der MRTD Chip in der Lage sein, seine "Trust-Points" intern zu aktualisieren, falls er ein gültiges CVCA Link Zertifikat. Dies ist nötig, da das Schlüsselpaar des CVCA mit der Zeit gewechselt wird.

Zweitens hat der Chip keine interne Uhr. Er hat ein Feld ("current date"), mit dem er die Zertifikate auf ihre Gültigkeit überprüft. Diese Feld wird in der Produktionsphase mit dem Datum der Erstellung initialisiert.

Sobald die Karte in Betrieb ist, aktualisiert der Chip das Feld. Jedes mal, wenn er ein gültiges Zertifikat erhält, vergleicht er am Ende, ob das Erstellungsdatum des Zertifikates jüngerem Ursprungs ist als sein "current date"-Feld. Ist

dies der Fall, ersetzt er den Wert des Feldes mit dem Erstellungsdatum des Zertifikates.

4.7.1 Country Verifying CAs

Jedes Bundesland muss ein Country Verifying CA, einen sogenannten “Trust-Point“ bereitstellen, welcher Document Verifier Zertifikate erstellt und herausgibt.

In diesen Zertifikaten legt das CVCA fest, welche Zugriffsrechte das Zertifikat hat und für welchen Zeitraum es ausgestellt wird. Um das Sicherheitsrisiko so gering wie möglich zu halten, werden die Zertifikate nur mit geringen Gültigkeitszeiträumen ausgestattet.

4.7.2 Document Verifier

Document Verifier sind Organisationen, wie beispielsweise die Landespolizei. Diese Organisationen sind, von mindestens einer CVCA, dazu authorisiert, Zertifikate für ihre eigenen Terminals zu erstellen. Diese Zertifikate können jedoch maximal die gleichen Rechte wie die Organisation selbst besitzen und können maximal ebenso lange gültig sein.

Falls ein Document Verifier Daten einer Karte eines anderen Bundeslandes erhalten möchte, muss er sich ein Document Verifier Zertifikat dieses Bundeslandes besorgen.

Kapitel 5

eID Ablauf und Protokolle im Detail

Dieses Kapitel wurde von Jan Reubold verfasst.

5.1 Ablauf der Identifikation

Die eID Funktion erlaubt elektronischen Zugriff auf alle persönlichen Daten ausgenommen der biometrischen Merkmale wie z.B. Fingerabdrücke, daher ist es wichtig, dass sie vor unerlaubten Zugriffen geschützt wird. Zu diesem Zweck muss *PACE (Password Authenticated Connection Establishment)*, *Terminal Authentication* und *Chip Authentication* ausgeführt werden, bevor man befugt ist die persönlichen Daten auszulesen.

Anhand von PACE wird eine sichere Verbindung aufgebaut, damit es einem Angreifer nicht möglich ist die Kommunikation zu belauschen. Nach PACE wird Secure Messaging (siehe unten) benutzt um *Terminal Authentication* und *Chip Authentication* durchzuführen.

Um herauszufinden welche Informationen das Terminal befugt ist auszulesen, und ob es überhaupt das Terminal ist, für welches es sich ausgibt, muss sich nun das Terminal der Karte gegenüber authentifizieren. Hierzu schickt das Terminal der Karte eine Reihe von Zertifikaten, welche die Karte verifiziert und aus diesen den öffentlichen Schlüssel des Terminals ausliest. Das Terminal authentifiziert sich nun mit einer signierten Nachricht, welche die Karte mit Hilfe des vorher erhaltenen öffentlichen Schlüssels überprüft. Während dieses Schrittes erstellt das Terminal außerdem ein vorübergehendes Diffie-Hellman Schlüsselpaar und schickt der Karte eine komprimierte Version des öffentlichen Schlüssels dieses Paares, dieser wird im nachfolgenden *Chip Authentication* benötigt.

Der Terminal muss nun seinerseits sichergehen, dass sein Gegenüber(die Karte) die Karte ist, für welche diese sich ausgibt. Hierfür erhält das Terminal von der Karte ein signiertes Datenobjekt (Signed Data), worin der öffentliche Schlüssel der Karte enthalten ist. Das Terminal schickt seinerseits den, während der *Terminal Authentication* erstellten, öffentlichen Schlüssel an die Karte. Nun können beide das Diffie-Hellman KeyAgreement ausführen, die Karte mit ihrem geheimen Schlüssel und dem gerade Erhaltenen, das Terminal mit seinem Geheimen und dem öffentlichen Schlüssel, welcher in dem signierten Datenob-

jekt enthalten war. Zur Authentifizierung berechnet die Karte nun den MAC über eine Nachricht mit dem gemeinsamen Geheimnis als Schlüssel und schickt ihn an das Terminal. Dieses berechnet seinerseits den MAC der Nachricht und überprüft somit die Authentizität der Karte.

5.2 PACE

PACE (Password Authenticated Connection Establishment) baut die initiale Verbindung und Verschlüsselung auf. Es verwendet hierbei ein, vom Nutzer eingegebenes Passwort (siehe unten) und, je nach Wunsch und Karte, entweder das Diffie-Hellman-, oder das Elliptic Curve Diffie-Hellman Protokoll.

Die folgende Beschreibung bezieht sich auf die Anwendungsschicht des PACE-Protokolls.

Falls man das Elliptic Curve Diffie-Hellman Protokoll verwendet, müssen zunächst die benötigten Domain Parameter von der Karte gelesen werden. Diese befinden sich in einer Datei Namens EF.CardAccess.

Eine genauere Beschreibung zum Auslesen der EF.CardAccess Datei und der Domain Parameter findet man weiter oben.

Die konkreten APDUs für diese und andere Operationen befinden sich am Ende dieses Kapitels.

Vor dem eigentlichen Authentifizierungsvorgang muss der Karte nun, anhand einer MSE (Manage Security Environment) Nachricht (siehe unten), mitgeteilt werden, welches Protokoll mit welcher Umgebung ausgeführt werden soll.

OID	Sym. Cipher	Key Length	Secure Messaging	Auth. Token
0.4.0.127.0.7.2.2.4.1.1	3DES	112	CBC / CBC	CBC
0.4.0.127.0.7.2.2.4.1.2	AES	128	CBC / CMAC	CMAC
0.4.0.127.0.7.2.2.4.1.3	AES	192	CBC / CMAC	CMAC
0.4.0.127.0.7.2.2.4.1.4	AES	256	CBC / CMAC	CMAC

Tabelle 5.1: *PACE mit Diffie-Hellman*

In dieser Nachricht wird, durch einen OID, unter anderem festgelegt, welches Protokoll ausgeführt werden soll, im Falle von PACE, entweder PACE mit DH oder ECDH (siehe Tabelle).

Desweiteren wird festgelegt welches Passwort für die Authentifizierung gewählt wurde (CAN oder PIN).

Nun kann der Authentifizierungsvorgang beginnen, er wird durch eine Reihe von General Authenticate (GA) Kommandos (siehe unten) durchgeführt.

OID	Sym. Cipher	Key Length	Secure Messaging	Auth. Token
0.4.0.127.0.7.2.2.4.2.1	3DES	112	CBC / CBC	CBC
0.4.0.127.0.7.2.2.4.2.2	AES	128	CBC / CMAC	CMAC
0.4.0.127.0.7.2.2.4.2.3	AES	192	CBC / CMAC	CMAC
0.4.0.127.0.7.2.2.4.2.4	AES	256	CBC / CMAC	CMAC

Tabelle 5.2: *PACE mit Elliptic Curve Diffie-Hellman*

Insgesamt werden vier GAs verschickt, die Erste dient lediglich als Initiator, worauf eine, mit dem Passwort verschlüsselte Nonce von der Karte als Antwort zurückgeschickt wird.

Im zweiten GA findet der erste Diffie-Hellman Schlüsselaustausch statt.

Das Terminal, sowie die Karte, erstellen jeweils ein vorübergehendes Diffie-Hellman Schlüsselpaar und schicken den dazugehörigen öffentlichen Schlüssel an ihr Gegenüber.

Nun können beide das Diffie-Hellman KeyAgreement durchführen. Jeweils mit dem privaten Schlüssel des eben erstellten Schlüsselpaares und dem vom Gegenüber zugeschickten öffentlichen Schlüssel.

Sobald sie das gemeinsame Geheimnis ausgerechnet haben, mappen beide die, in den Domain Parametern enthaltene, Basis G:

- $G' = sG + P$ bei PACE mit Elliptic Curve Diffie-Hellman, wobei s die Nonce ist und $P = xY = yX$ das gemeinsame Geheimnis
- $G' = G^s + P$ bei PACE mit Diffie-Hellman, wobei s die Nonce ist und $P = Y^x = X^y$ das gemeinsame Geheimnis

Mit dieser neuen Basis G' wird im nächsten Schritt der zweite Diffie-Hellman Schlüsselaustausch berechnet. Erneut erstellen beide Protagonisten einen vorübergehendes Diffie-Hellman Schlüsselpaar und schicken den öffentlichen Schlüssel des jeweiligen Schlüsselpaares an den Gegenüber.

Nun führen beide ein weiteres Mal das Diffie-Hellman KeyAgreement durch. Das daraus gewonnene gemeinsame Geheimnis K wird nun als Schlüsselmaterial für die KDF benötigt, um einen MAC Schlüssel K_{MAC} daraus abzuleiten.

Zur Kontrolle wird im letzten GA Schritt der MAC des gesamten Public Key Data Objekts (siehe unten) mit dem MAC Schlüssel K_{MAC} berechnet. Es wird schließlich zum Vergleichen an die Karte geschickt, welche ihrerseits den MAC über ein ähnliches Public Key Data Objekt berechnet (nur der öffentliche Schlüssel unterscheidet sich von dem der Version des Terminals) und diesen zurückschickt. Falls erfolgreich kann nun, mit den vorher berechneten K_{MAC} und K_{Enc}, Secure Messaging gestartet werden.

5.2.1 Ablauf von PACE

Die Karte(PICC)	$\rightleftharpoons$	Das Terminal(PCD)
wähle zufälliges NONCE s aus		
$K_\pi = KDF_\pi(\pi)$		$K_\pi = KDF_\pi(\pi)$
$z = ENC(K_\pi, s)$		
	$\xrightarrow{z}$	
		$s = DEC(K_\pi, z)$
		wähle ein zufälliges x_1
		ECDH: $X_1 = x_1 * G$
		DH: $X_1 = G^{x_1}$ mod p
	$\xleftarrow{X_1}$	
abbruch, falls X_1 ungültig		
wähle ein zufälliges y_1		
ECDH: $Y_1 = y_1 * G$		
DH: $Y_1 = G^{y_1}$ mod p		
	$\xrightarrow{Y_1}$	
		abbruch, falls Y_1 ungültig
ECDH: $P = y_1 * X_1$		ECDH: $P = x_1 * Y_1$
DH: $P = X_1^{y_1}$ mod p		DH: $P = Y_1^{x_1}$ mod p

——— Abschluß des 1. (EC)DH KeyAgreements (gemeinsamens Geheimnis P) ———

ECDH: $G' = s * G + P$	ECDH: $G' = s * G + P$
DH: $G' = G^s + P$ mod p	DH: $G' = G^s + P$ mod p
	wähle ein zufälliges x_2
	ECDH: $X_2 = x_2 * G'$
	DH: $X_2 = G'^{x_2}$ mod p
$\xleftarrow{X_2}$	
abbruch, falls X_2 ungültig	
wähle ein zufälliges y_2	
ECDH: $Y_2 = y_2 * G'$	
DH: $Y_2 = G'^{y_2}$ mod p	
$\xrightarrow{Y_2}$	
	abbruch, falls Y_2 ungültig
ECDH: $K = y_2 * X_2$	ECDH: $K = x_2 * Y_2$
DH: $K = X_2^{y_2}$ mod p	DH: $K = Y_2^{x_2}$ mod p
—— Abschluß des 2. (EC)DH KeyAgreements (gemeinsames Geheimnis K) ——	
$K_{MAC} = KDF_{MAC}()$	$K_{MAC} = KDF_{MAC}()$
	$t_{PCD} = MAC(k_{MAC}, (Y_2, D))$
$t_{PICC} = MAC(k_{MAC}, (X_2, D))$ $\xleftarrow{t_{PCD}}$	
$\xrightarrow{t_{PICC}}$	
$t'_{PCD} = MAC(k_{MAC}, (Y_2, D))$	$t'_{PICC} = MAC(k_{MAC}, (X_2, D))$
abbruch, falls $t_{PCD} \neq t'_{PCD}$	abbruch, falls $t_{PICC} \neq t'_{PICC}$

5.2.2 Public Key Data Objekt

Das Public Key Data Objekt enthält alle Informationen sowohl - falls Elliptic Curve Diffie-Hellman verwendet wird - über die elliptische Kurve, als auch über das vereinbarte Geheimnis:

Diffie-Hellman Public Keys

Datenobjekt	Abkürzung	Tag	Typ
Object Identifier		0x06	Object Identifier
Prime modulus	p	0x81	Unsigned Integer
Order of the subgroup	q	0x82	Unsigned Integer
Generator	g	0x83	Unsigned Integer
Public value	y	0x84	Unsigned Integer

Tabelle 5.3: Public Key Data Objekt bei Diffie-Hellman

Public Key Data: $\mathbf{PK} = \{\text{oid}, p, q, g, y\}$ wobei alle Parameter ausser y identisch zu den entsprechenden statischen Domain Parametern sind.

Elliptic Curve Diffie-Hellman Public Keys

Datenobjekt	Abkürzung	Tag	Typ
Object Identifier		0x06	Object Identifier
Prime modulus	p	0x81	Unsigned Integer
First coefficient	a	0x82	Unsigned Integer
Second coefficient	b	0x83	Unsigned Integer
Base Point	G	0x84	Elliptic Curve Point
Order of the base point	r	0x85	Unsigned Integer
Public point	Y	0x86	Elliptic Curve Point
Cofactor	f	0x87	Unsigned Integer

Tabelle 5.4: Public Key Data Objekt bei Elliptic Curve Diffie-Hellman

Public Key Data: $\mathbf{PK} = \{\text{oid}, p, a, b, n, h, G', X_2\}$ wobei alle Parameter ausser G' (Base point) und Y (Public Point) identisch zu den entsprechenden statischen Domain Parametern sind.

Das gesamte Public Key Data Objekt ist TLV-codiert (Tag - Length - Value [4]). Die oben aufgeführten Informationen sind das Value des Objekts. In beiden Fällen, mit ECDH und mit DH, ist das Tag 0x7F49.

Das Feld Length gibt die Länge des darauffolgenden Value-Feldes an. In diesem Fall also die Länge der, in der Tablle aufgeführten, Informationen. Sollte die Länge kleiner als 128 Bytes sein, wird einfach die Länge als Hexadezimal-Zahl angegeben. Sollte sie jedoch größer sein, wird ein weiteres Tag der Länge vorangestellt. Entweder 0x81, wenn die Länge größer 127, jedoch kleiner 256 Bytes ist, oder 0x82 wenn sie größer als 255 Bytes ist.

5.2.3 Verwendbare Passwörter

Für PACE kann man vier verschiedene Passwörter verwenden, die CAN (Card Access Number), die PIN (Personal Identification Number), den PUK (Pin Unblock Key) und das MRZ-Password (Machine Readable Zone Password).

CAN

Die CAN ist eine Nummer auf dem Chip, welche von jedem abgelesen werden kann, der visuellen Zugriff auf den Chip hat. Sie kann statisch auf den Chip gedruckt sein oder auch dynamisch auf einem Display wiedergegeben werden. Die CAN ist relativ groß, sie kann beliebig oft (falsch) eingegeben werden ohne zu blockieren.

PIN

Die PIN ist eine geheime Nummer, welche nur der Ausweisinhaber kennt. Sie ist 6-stellig und somit nicht resistent gegen Brute Force Angriffe, die PIN blockiert nach drei Fehlversuchen und muss mit der PUK reaktiviert werden.

PUK

Der PUK ist ein langes sicheres Passwort, welches nur der Eigentümer des Chips kennt. Der PUK ist groß genug um gegen Brute Force Angriffe gerüstet zu sein, man kann ihn also beliebig oft (falsch) eingeben, ohne zu blockieren. Verwendet man den PUK, erhält man die Möglichkeit die PIN zu entsperren, jedoch keine weiteren Funktionalitäten.

MRZ-Password

Mit dem MRZ-Password hat man nur Zugriff auf die ePassport Funktionen. Es ist ein statisches Password, das sich von der Machine Readable Zone ableitet. Es kann nur benutzt werden, wenn die MRZ auf dem MRTD gedruckt ist.

Somit erlauben sowohl CAN als auch PIN Zugriff auf die eID Funktion, der Hintergrund erklärt sich wie folgt:

Eine Person, welche sich im Besitz des Ausweises befindet, braucht sich nicht zu authentifizieren um die persönlichen Daten, die auf dem Chip enthalten sind, lesen zu können (CAN wird benutzt). Falls aber der Chip aus der Entfernung gelesen wird, beispielsweise Online, so reicht der Besitz des Ausweises zur Authentifizierung nicht aus. In diesem Fall sollte die PIN verlangt werden, die CAN würde die Authentizität nicht erhöhen, da sie ohnehin auf dem Ausweis sichtbar und somit nicht nur dem Inhaber bekannt ist. Dies entspricht der vorherigen Situation eines Ausweises mit ausschließlich visuellen Merkmalen, aber mit der zusätzlichen Möglichkeit einer freiwilligen Authentifizierung ohne direkten Zugriff auf den Ausweis.

5.2.4 PACE APDUs

Select Master File: "00 A4 00 0C 02 3F 00"

Select CardAccess File: "00 A4 02 0C 02 01 1E"

Read CardAccess File: "00 B0 xx 00 xx"

Das xx steht für den entsprechenden Schritt, 00 für die ersten 256 Byte, 01 die nächsten usw. Das 'xx' am Ende steht für die expected length, '00' für beliebig.
Antwort: DER encoded Data.

MSE: "00 22 C1 A4 0F 80 0A 04 00 7F 00 07 02 02 04 02 01 83 01 03"

Nach der CLA/INS wählen wir mit '80' den OID für das gewünschte Protokoll (z.B. 0.4.0.127.0.7.2.2.4.2.1), dieser wird jedoch noch wie folgt transformiert.
Die ersten beiden Werte des OID, in diesem Fall 0 und 4, werden mit einer Formel zusammengerechnet. Der neue Wert ergibt sich auch dem 40-fachen des ersten Wertes addiert mit dem zweiten Wert.
In diesem Fall also $40 * 0 + 4 = 4$.
Die restlichen Werte bleiben gleich

Danach folgt eine '83' als Indikator für das Passwort, dass benutzt werden soll.
Hierbei können folgende Werte benutzt werden:
0x01: MRZ
0x02: CAN
0x03: PIN
0x04: PUK

General Authenticate:

GA(1): "10 86 00 00 02 7C 00 xx"

Das Byte '7C' ist vorgeschrieben für alle GAs, das folgende null Byte muss explizit für den leeren Inhalt angegeben werden, das 'xx' steht für die erwartete Länge, in diesem Falle der nonce, möglich sind '22' (oder höher) oder '00' für beliebig (256).
Antwort: verschlüsselte Zufallszahl (nonce).

GA(2): "10 86 00 00 xx 7C xx 81 xx"+ EC Point + expected Length

Die xx Bytes sind die Lc Bytes (Content Length), welche variiert. Der EC point ist der zufällige Punkt (öffentliche Schlüssel) auf der elliptischen Kurve, repräsentiert als Octet String mit einer uncompressed Kodierung.
Antwort: der zufällige Punkt der Karte, ebenfalls als Octet String in der gleichen Kodierung.

GA(3): "10 86 00 00 xx 7C xx 83 xx"+ EC Point + expected Length

'xx' ist wieder die Content Length. Der EC Point wurde diesmal mit dem neuen Basepoint berechnet und wird in der gleichen Octet String Kodierung verschickt.

Antwort: zufälliger Punkt mit dem gleichen neuen Basepoint

GA(4): "10 86 00 00 xx 7C xx 85 xx"+ MAC Token + expected Length

Wie vorher steht 'xx' für die jeweilige Content Length, da der MAC eine statische Länge von 8 Bytes hat betragen diese immer 0C, 0A und 08. Als expected length kann 00 gewählt werden. Das Token wird zum Vergleichen an die Karte geschickt.

Antwort: Die Karte schickt ihr Token zurück, falls man dies überprüfen möchte.

5.3 Terminal Authentication

Terminal Authentication (TA) wird direkt nach PACE durchgeführt und dient der Authentifizierung des Terminals gegenüber der Karte. Das Terminal benutzt als erstes hierbei eine Reihe von Zertifikaten, eine Zertifikatskette. Diese Kette ist so aufgebaut, dass die jeweils höhere Instanz ein Zertifikat für die darunterliegende ausstellt. Die höchste Instanz dieser Zertifikate ist der Karte bekannt und kann überprüft werden, die letzte Instanz ist das Terminal selbst.

Die Karte entnimmt, nach erfolgreicher Verifikation der Zertifikate, dem letzten Zertifikat den öffentlichen Schlüssel des Terminals.

In der von uns implementierten Version, erstellt das Terminal nun ein weiteres vorübergehendes Diffie-Hellman Schlüsselpaar. Es wird für die darauffolgende *Chip Authentication* benötigt, also schickt das Terminal den öffentlichen Schlüssel dieses Schlüsselpaares in einer komprimierten Version an die Karte.

In diesem Schritt muss der Terminal außerdem zusätzliche Daten ("auxiliary data") mitschicken, falls bei späteren Operationen, die der Terminal ausführt, diese Daten benötigt werden. Wenn beispielsweise eine Altersverifikation durchgeführt werden soll, muss der Terminal das Geburtsdatum mitschicken.

Der öffentliche Schlüssel, sowie die optionalen zusätzlichen Daten, werden per MSE an die Karte geschickt.

Hierbei wird auch der zu verwendende Algorithmus für die Signatur festgelegt. Es gibt zwei Möglichkeiten, entweder RSA (Rivest, Shamir, Adleman) oder ECDSA (Elliptic Curve Digital Signature Algorithm) (siehe unten).

Sobald das abgeschlossen ist, schickt das Terminal der Karte eine sogenannte "Challenge". Die Karte erstellt eine Zufallszahl r_{PICC} und schickt diese an das Terminal zurück.

Das Terminal authentifiziert sich nun, indem es eine Signatur über:

- dem PACE-Token (ID_{PICC}, dem öffentlichen Schlüssel der Karte vom finalen Diffie-Hellman KeyAgreement in PACE),
- der Zufallszahl (r_{PICC}),
- der komprimierten Version des eben erstellten, vorübergehenden, öffentlichen Schlüssel,
- eventuellen zusätzlichen Daten

mit seinem privaten Schlüssel und dem oben festgelegten Algorithmus erstellt.

Nach Erhalt der Signatur verifiziert die Karte diese, und damit auch die Authentizität des Terminals, mit dem öffentlichen Schlüssel des Terminals.

Bei erfolgreicher Authentifizierung ist somit bewiesen, dass das Terminal mit der gültigen Signatur im Besitz aller übertragenen Daten, sowie des, zu dem zertifizierten öffentlichen Schlüssel gehörenden, privaten Schlüssels ist.

5.3.1 Terminal Authentication mit RSA

Für Terminal Authentication mit RSA müssen folgende Algorithmen verwendet werden:

OID	Signature	Hash	Parameter
0.4.0.127.0.7.2.2.2.1.1	RSASSA-PKCS1-v1.5	SHA-1	N/A
0.4.0.127.0.7.2.2.2.1.2	RSASSA-PKCS1-v1.5	SHA-256	N/A
0.4.0.127.0.7.2.2.2.1.3	RSASSA-PSS	SHA-1	default
0.4.0.127.0.7.2.2.2.1.4	RSASSA-PSS	SHA-256	default

Tabelle 5.5: *Terminal Authentication mit RSA*

5.3.2 Terminal Authentication mit ECDSA

Für Terminal Authentication mit ECDSA müssen folgende Algorithmen verwendet werden:

OID	Signature	Hash
0.4.0.127.0.7.2.2.2.2.1	ECDSA	SHA-1
0.4.0.127.0.7.2.2.2.2.2	ECDSA	SHA-224
0.4.0.127.0.7.2.2.2.2.3	ECDSA	SHA-256

Tabelle 5.6: *Terminal Authentication mit ECDSA*

5.3.3 Ablauf von Terminal Authentication

Die Karte(PICC)	$\rightleftharpoons$	Das Terminal(PCD)

Zu Beginn schickt das Terminal die Zertifikate seiner Zertifikatskette einzeln, mit den Befehlen MSE:Set DST und PSO:Verify Certificate (siehe unten), an die Karte. Diese verifiziert die Zertifikate und extrahiert aus dem Zertifikat des Terminals dessen öffentlichen Schlüssel.

$PK_{PCD} = extract(Cert)$

wähle ein zufälliges x_1

ECDH: $X_1 = x_1 * G$

DH: $X_1 = G^{x_1}$ mod p

$\widetilde{cPK}_{PCD} = Comp(X_1)$

$\xleftarrow{MSE{:}SetAT(\widetilde{cPK}_{PCD})}$

wähle ein zufälliges r_{PICC} aus

$\xrightarrow{r_{PICC}}$

$m = ID_{PICC}||r_{PICC}||\widetilde{cPK}_{PCD}||A_{PCD}$

$s_{PCD} = Signatur(SK_{PCD}, m)$

$\xleftarrow{s_{PCD}}$

$m = ID_{PICC}||r_{PICC}||\widetilde{cPK}_{PCD}||A_{PCD}$

$verify(PK_{PCD}, s_{PCD}, m)$

5.4 Chip Authentication

Chip Authentication (CA) wird nach Terminal Authentication (TA) durchgeführt, wobei der MRTD Chip sich in diesem Schritt bei dem Terminal authentifiziert.

Zu Beginn teilt das Terminal per MSE der Karte mit, welches Protokoll mit welchen Algorithmen ausgeführt werden soll. Wie in PACE kann man Chip Authentication mit Diffie-Hellman, aber auch mit Elliptic Curve Diffie-Hellman ausführen (siehe unten).

OID	Sym. Cipher	Key Length	Secure Messaging	Auth. Token
0.4.0.127.0.7.2.2.3.1.1	3DES	112	CBC / CBC	CBC
0.4.0.127.0.7.2.2.3.1.2	AES	128	CBC / CMAC	CMAC
0.4.0.127.0.7.2.2.3.1.3	AES	192	CBC / CMAC	CMAC
0.4.0.127.0.7.2.2.3.1.4	AES	256	CBC / CMAC	CMAC

Tabelle 5.7: *Chip Authentication mit Diffie-Hellman*

OID	Sym. Cipher	Key Length	Secure Messaging	Auth. Token
0.4.0.127.0.7.2.2.3.2.1	3DES	112	CBC / CBC	CBC
0.4.0.127.0.7.2.2.3.2.2	AES	128	CBC / CMAC	CMAC
0.4.0.127.0.7.2.2.3.2.3	AES	192	CBC / CMAC	CMAC
0.4.0.127.0.7.2.2.3.2.4	AES	256	CBC / CMAC	CMAC

Tabelle 5.8: *Chip Authentication mit Elliptic Curve Diffie-Hellman*

Danach liest das Terminal, wie in PACE schon beschrieben, die benötigten Daten, einen statischen, öffentlichen Diffie-Hellman Schlüssel PK_{PICC} und , falls benötigt, die Domainparameter D_{PICC}, aus der EF.CardAccess Datei aus.

Das Terminal schickt daraufhin den, in *Terminal Authentication* erstellten, vorübergehenden öffentlichen Schlüssel an die Karte.

Die Karte überprüft nun erst einmal, ob das Terminal noch das Terminal ist, das sich im vorherigen Schritt authentifiziert hat. Es komprimiert den eben erhaltenen, vorübergehenden, öffentlichen Schlüssel nämlich und vergleicht ihn mit dem, während der *Terminal Authentication* erhaltenen, vorübergehenden Schlüssel.

Da das Terminal nun den öffentlichen Schlüssel der Karte hat und die Karte den vorübergehenden öffentlichen Schlüssel des Terminals, können beide mit den jeweiligen privaten Schlüsseln das Diffie-Hellman KeyAgreement durchführen und erhalten das gemeinsame Geheimnis K.

Die Karte leitet sich nun, mit dem gemeinsamen Geheimnis K und einer zufällig ausgewählten Zahl r_{PICC}, die Schlüssel K_{MAC} und K_{Enc} ab und erstellt einen MAC über ein Public Key Data Objekt mit dem Schlüssel K_{MAC}.

Die Karte schickt die Zufallszahl r_{PICC} und das Token (den MAC) an das Terminal, welches nun seinerseits die Schlüssel K_{MAC} und K_{Enc} ableiten kann

und den MAC über das gleiche Public Key Data Objekt berechnet um es mit dem Token zu vergleichen.

Damit hätte sich der MRTD Chip noch nicht authentifiziert, da alle Informationen letztlich aus dem Chip stammen und somit gefälscht werden könnten. Der öffentliche Diffie-Hellman Schlüssel der Karte, welcher im ersten Schritt ausgelesen wurde, muss nun noch mit Passive Authentication verifiziert werden.

5.4.1 Ablauf von Chip Authentication

Die Karte(PICC)	$\rightleftharpoons$	Das Terminal(PCD)

Das Terminal extrahiert aus der EF.CardSecurity Datei (siehe oben) den öffentlichen Schlüssel (Y_1) der Karte und, falls benötigt, die CA Domain Parameter (D_{PICC}).

$\widetilde{cPK}_{PCD}$ aus der TA

X_1 und x_1 aus der TA

$\xleftarrow{X_1}$

abbruch, falls $\widetilde{cPK}_{PCD} \neq Comp(X_1)$

ECDH: $K = y_1 * X_1$

ECDH: $K = x_1 * Y_1$

DH: $K = X_1^{y_1} \bmod p$

DH: $K = Y_1^{x_1} \bmod p$

——— Abschluß des (EC)DH KeyAgreements (gemeinsamens Geheimnis K) ———

wähle zufälliges NONCE r_{PICC}

$K_{MAC} = KDF_{MAC}(K, r_{PICC})$

$K_{ENC} = KDF_{ENC}(K, r_{PICC})$

$T_{PICC} = MAC(K_{MAC}, (X_1, D_{PICC}))$

$\xrightarrow{r_{PICC}, T_{PICC}}$

$K_{MAC} = KDF_{MAC}(K, r_{PICC})$

$K_{ENC} = KDF_{ENC}(K, r_{PICC})$

$T'_{PICC} = MAC(K_{MAC}, (X_1, D_{PICC}))$

abbruch, falls $T_{PICC} \neq T'_{PICC}$

5.5 Passive Authentication

Auf dem MRTD Chip befindet sich eine Signatur, welche bei der Personalisierungsphase des Chips erstellt wurde und einen Hash über alle Datengruppen (Domain Parameters, Name etc...), die auf dem Chip vorhanden sind, enthält.

Die Passive Authentication(PA) kann somit manipulierte Datengruppen erkennen. Um dies zu machen, muss der Terminal folgende Schritte ausführen:

1. das Security Object des MRTD-Chips auslesen
2. Herausholen des dazugehörigen Document Signer Certificate, dem zu vertrauenden Country Signing CA Certificate und der Certificate Revocation List(CRL)
3. Verifizierung des Document Signer Certificate und der Signatur des Security Object
4. Berechnung des Hashwertes der einzelnen Datengruppen und Vergleich mit den Hashwerten in dem Security Object

5.6 Secure Messaging

Secure Messaging ist ein sicherer Tunnel, in dem APDUs verschlüsselt verschickt werden können. Die APDUs werden dabei in einen verschlüsselten „Mantel“ gekleidet und schließlich auf der anderen Seite entschlüsselt.

Secure Messaging benutzt zwei Schlüssel: k_{Enc} und k_{MAC}, beide aus der KDF abgeleitet und in PACE oder CA berechnet. k_{Enc} wird benutzt um diverse Datenobjekte zu verschlüsseln, k_{MAC} wird nur für den MAC benutzt.

Padding sowohl in dem Datenblock als auch bei der Berechnung des MAC richtet sich nach dem Standard ISO 7816-4.

5.6.1 CAPDU Transformation

Der APDU Header einer zu verschlüsselnden CAPDU wird durch die Verschlüsselung nicht verändert, jedoch muss das Klassenbyte 0xXC betragen, wobei X ein funktionsspezifischer Parameter ist (die niederwertigsten 4 Bit in dem Byte müssen C (1010) ergeben).

Generell erzeugt Secure Messaging diskrete Datenobjekte, verschlüsselt in DER, und setzt diese wieder in die APDU ein, mit einem entsprechenden neuen Lc' Parameter für die neue Länge der Daten in der gesicherten CAPDU. Jedes einzelne Datenobjekt entspricht einer Verschlüsselung der Objekte in der ursprünglichen APDU.

So wird der Daten Teil der APDU in gleichgroße Blöcke unterteilt, mit Padding wird die Gesamtlänge an die Blocklänge angepasst, und die einzelnen Blöcke werden mit k_{Enc} entweder in TDES oder AES Modus verschlüsselt.

Der Le Parameter wird unverschlüsselt mit dem Tag $'97'$ versehen und der gesamte Block mit Header, den formatierten Daten, dem Le Parameter und dem zugehörigen Padding wird mit einer MAC Funktion gehasht. Die resultierende

APDU ist wie folgt aufgebaut: Zunächst der unveränderte Header, der neue Lc' Parameter, die formatierten Daten, angeführt von einem $'87'$ Tag, der Länge und $'01'$, gefolgt von dem Le Parameter, dem berechneten MAC und einem neuen Le' Parameter $'00'$.

Falls Parameter in der APDU fehlen, wird der entsprechende Teil ausgelassen. Beispielsweise würde eine leere APDU, bestehend ausschließlich aus dem Header, folglich nur aus dem Header, Lc', dem MAC aus Header und Padding und Le' bestehen.

Die nachfolgende Abbildung stellt einen Ablauf für eine vollständige APDU inklusive Datenblock und Le Parameter dar:

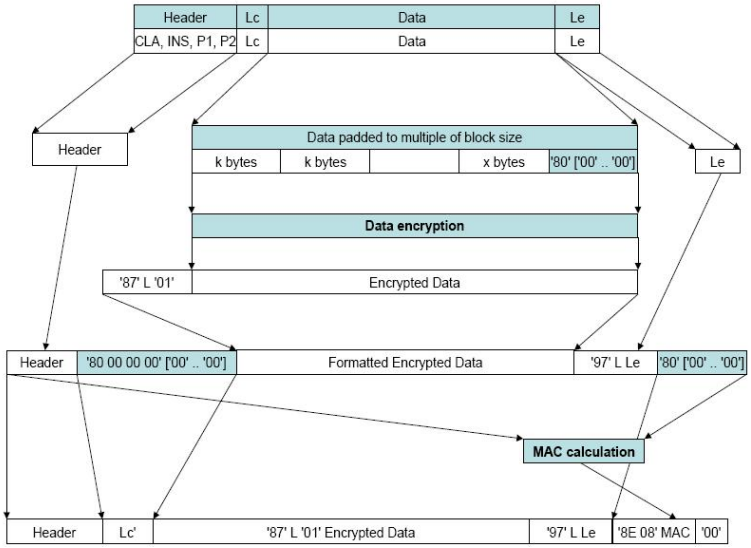

Abb. 5.1: *Darstellung einer CAPDU Transformation für Secure Messaging; Quelle: BSI-Technische Richtlinien*

5.6.2 RAPDU Transformation

Die Antwort auf eine gesicherte CAPDU wird nach dem gleichen Prinzip transformiert, das Statuswort bleibt am Ender der RAPDU erhalten, die Daten werden ebenfalls verschlüsselt und mit dem Statuswort und Padding in den MAC eingebracht.

Die transformierte RAPDU besteht somit aus dem verschlüsselten Datenfeld, angeführt mit einer ′87′, dem Statuswort angeführt mit ′9902′, dem MAC und dem unveränderten Statuswort, wie in der folgenden Abbildung dargestellt.

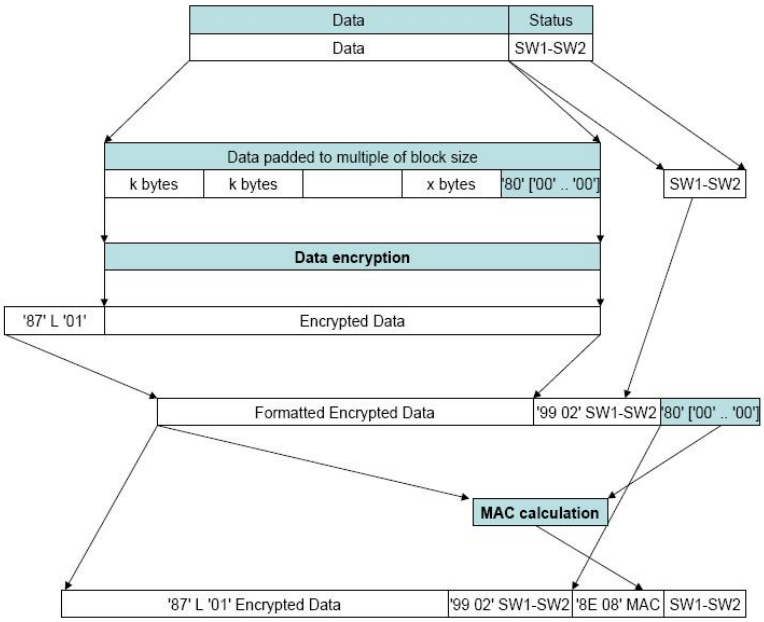

Abb. 5.2: *Darstellung einer RAPDU Transformation für Secure Messaging; Quelle: BSI-Technische Richtlinien*

5.7 Die Nachrichten

Hier wird ein kurzer Überblick über die einzelnen Nachrichten-Typen der Protokolle gegeben.

Die MSE:Set AT und die GA sind die Pfeiler der meisten Protokolle. Anhand dieser Nachrichten werden die gewünschten Protokolle vereinbart (MSE:Set AT) und ausgeführt (GA).

Manchmal werden noch weitere Nachrichten-Typen benötigt, von denen einige hier auch behandelt werden.

5.7.1 ManageSecurityEnvironment:Set AT

MSE ist ein allgemeines Konstrukt, mit dessen Hilfe der Karte mitgeteilt werden kann, welches Protokoll in welcher Umgebung ausgeführt werden soll.

Insgesamt gibt es sieben verschiedene Informationstypen, von denen, je nach gewünschtem Protokoll, unterschiedliche in der MSE enthalten sind. Die Information über die OID des gewünschten Protokolls ist in jeder MSE enthalten.

5.7.2 GeneralAuthenticate

GA ist wie die MSE ein allgemeines Konstrukt, sprich es wird von PACE, ChipAuthentication und Restricted Identification, auf unterschiedliche Art und Weise, verwendet.

Es dient dazu die Nachrichten, die aufgrund des Protokolls zwischen Karte und Client ausgetauscht werden müssen, codiert zu verschicken.

5.7.3 ManageSecurityEnvironment:Set DST

Die MSE:Set DST Nachricht initialisiert die Verifikation eines Zertifikats. Es gibt eine Referenz zum öffentlichen Schlüssel des in der nachfolgenden PSO:Verify Certificate Nachricht enthaltenen Zertifikats an. MSE:Set DST und PSO:Verify Certificate ist eine miteinander verbundene Nachrichtenabfolge, auf eine MSE:Set DST folgt immer auch eine PSO-Nachricht.

5.7.4 PSO: Verify Certificate

Dieser Nachrichten-Typ dient einerseits dazu, die einzelnen Zertifikate zu verifizieren, sie gleichzeitig aber auch zu importieren. Mit diesem Befehl schickt das Terminal den Body und die Signatur jedes einzelnen seiner Zertifikate an die Karte. Diesem Nachrichten-Typ muss immer eine MSE:Set DST Nachricht vorangegangen sein.

Kapitel 6

Implementierung

In diesem Kapitel wird genauer auf das Design der Implementierung der, in den vorigen Kapiteln vorgestellten, Protokolle eingegangen. Da die Implementierung aus zwei Teilen besteht, aus der Client- sowie der Server-Implementierung, wird die Betrachtung auch in diese beiden Bereiche eingeteilt.

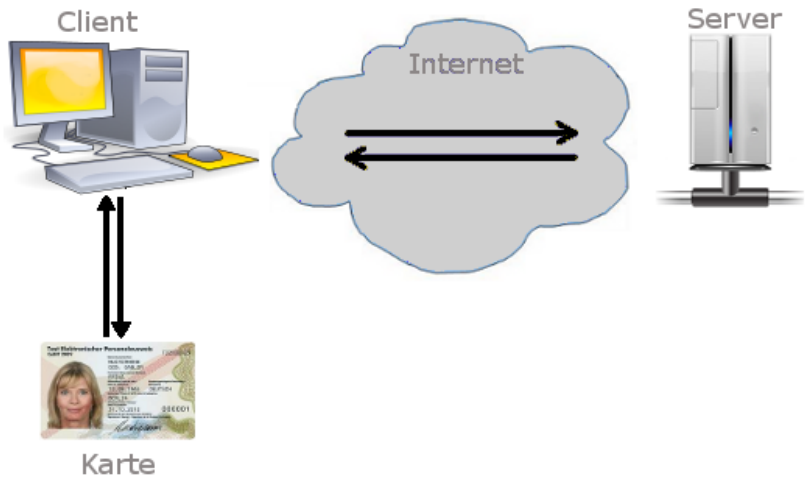

Abb. 6.1: Client-Server Architektur

Dieses Kapitel wurde von Jan Reubold verfasst.

6.1 Client

Der Client ist das bevorzugte Rechenorgan. Denn alles was sich hier berechnen lässt, muss nicht weiter an den Server und wieder zurück geschickt werden. Das erhöht die Performanz in dem es die Kommunikation zwischen Client und Server reduziert.

Er ist die Schnittstelle zwischen allen anderen Teilnehmern(Karte, User, Server). Für die Kommunikation zwischen der Karte und den anderen Teilnehmern ist er zusätzlich auch noch Übersetzer. Er wandelt z.B. Nachrichten vom Server in gültige APDUs um und schickt diese an die Karte, oder extrahiert aus den APDUs, die er von der Karte erhält, die benötigten Informationen und reicht diese, falls erwünscht, an den Server weiter. Der Client interagiert mit dem User, er fragt ihn z.B. nach dem PIN, zeigt ihm die Nutzungsbedingungen an oder lässt den User auswählen welche Informationen dieser an das Terminal übermitteln soll.

6.1.1 Design

Der Client ist unter anderem für die Abwicklung von PACE zuständig und muss somit mit jeder Version von PACE umgehen können.

Hierzu muss er mit ...

- ...*elliptischen Kurven* genau so, wie mit *unsignierten Integern* rechnen können
- ...er muss mit
  - *Triple Data Encryption Standard(Triple DES)* [16]
  - *Advanced Encryption Standard(AES)* [17]
  - *Electronic Code Book Mode(ECB)* [3]
  - *Cipher Block Chaining Mode(CBC)*
  - *Cipher based MAC(CMAC)* [19]

  rechnen und umgehen können
- ...und muss unter anderem den ...
  - ...SHA-1
  - ...SHA-256

  jeweils nach [18], berechen können.

Die erforderlichen Algorithmen werden mit Hilfe des JCE/JCA-Frameworks bereitgestellt. Es wurde darauf geachtet, dass der Client providerunabhängig ist, das heißt, dass es eine Schnittstelle gibt um neue Provider einzubinden und bestimmte Algorithmen mit diesem Provider zu verknüpfen.

Außerhalb dieser Schnittstelle wird, egal welcher Provider benutzt wird, einheitlich auf die Funktionen der verschiedenen Algorithmen zugegriffen.

Da der ePA sich zur Zeit noch in der Entwicklungsphase befindet und es ausserdem immer zu Veränderungen kommen kann, war es wichtig eine leichte Erweiterbarkeit der Funktionalität im Code zu ermöglichen.

Dies ist auch mit Blick auf die Zukunftsfähigkeit des Codes sinnvoll, da z.B. heute für sicher gehaltenen Hashfunktionen schnell als gebrochen dastehen können.

So musste eine Lösung gefunden werden, mit der man schnell und leicht bestimmte Algorithmen ersetzen kann. Dies bezog sich vor allem auf die Hashfunktionen und den Modus in dem das Diffie-Hellman KeyAgreement verwendet wird.

Wir entschieden uns das Strategie-Pattern (siehe unten) hierfür zu verwenden. Es stellte sich als kein Problem dar, die Algorithmen in Familien einzuteilen und zu kapseln.

So entstanden nun zwei Familien, die Familie der Hashfunktionen und die Familie der Diffie-Hellman Modi, mit jeweils zwei Mitgliedern. Die Anzahl der Familienmitglieder ist, mit jeweils zwei Mitgliedern relativ gering, aber um ein Erweiterung auf einfachem Wege zu ermöglichen, war es sinnvoll dieses Pattern anzuwenden.

Nach Anwendung des Pattern, musste ein weiteres Problem gelöst werden, die Lesbarkeit des Codes musste verbessert werden.

Dadurch, dass die beiden Strategie-Familien eine so große Fülle an Funktionen beherbergten war der Code für Fremdleser unlesbar. Der restliche Code sollte außerdem von den Strategie-Familien, so gut es eben geht, abgekapselt werden. Es sollte nur noch eine lose Koppelung bestehen.

Es musste eine Lösung gefunden werden, die dieses Problem beseitigen würde. Eine Lösung, die aus komplexen Funktionsaufrufsorgien einen primitiven Funktionsaufruf machen könnte.

Die Lösung war das Fassade-Pattern (siehe unten), dass allem Anschein nach all diese Probleme beseitigen könnte.

Die Fassaden-Klasse wurde Operator genannt.

Der Operator hat Zugriff auf alle Subklassenfunktionen und bietet auch einfache Funktionsaufrufe hinter denen sich viel komplexere Funktionsaufrufe verbergen.

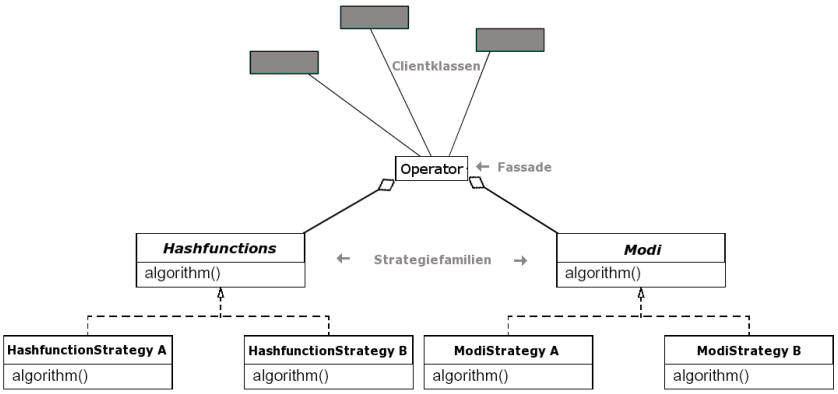

Abb. 6.2: Verschmelzung von Fassade- und Strategie-Pattern

Mit der Anwendung dieser Pattern ist unser Code zukunftsfähig und wartungsfreundlich. Er bietet die Möglichkeit, die komplizierten Vorgänge mancher Aufrufe auf unterster Ebene zu implementieren und diese von den Klassen, die diese Aufrufe tätigen zu entkoppeln.

Fassade-Pattern [9]

Falls man viele verschiedene Funktionen in verschiedenen Klassen zum Ausführen bestimmter Protokolle hat und falls man die ausführenden Klassen so gut wie möglich von der steuernden Klasse entkoppeln möchte, ist das Fassade-Pattern eine gute Wahl.

Dieses Pattern bietet eine einheitliche Schnittstelle zu einer Menge von Schnittstellen in Subklassen.

Diese Schnittstelle, die sogenannte “Fassade“, steuert alle ankommenden Anfragen und leitet diese an die zuständige Subklasse weiter. Die Fassade weiss also welche Subklasse für welche Anfrage zuständig ist.

Die Subklassen führen dann die ihnen zugewiesenen Aufgaben aus, wissen aber selbst nichts von der Fassade.

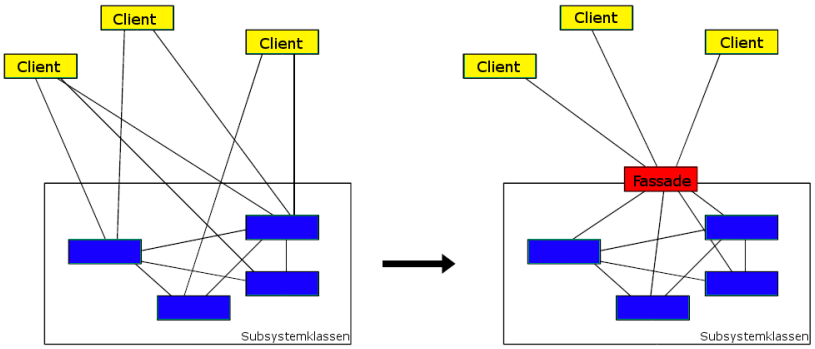

Abb. 6.3: Fassade-Pattern

Es reduziert die Anzahl der Objekte, die von der steuernden Klasse benutzt werden müssen.

Einzelne Komponenten der Subklassen sind oftmals stark aneinander gekoppelt. Das Fassade-Pattern ermöglicht eine lose Kopplung zwischen den Subsystemen und der steuernden Klasse.

Dies ermöglicht das problemlos Ersetzen von Subklassen, ohne dass die steuernde Klasse davon beeinträchtigt wird.

Strategie-Pattern

Es gibt oftmals viele verschiedene Wege (Algorithmen) um ein Problem zu lösen. Man denke nur an das Sortieren, eines der ersten Probleme, mit denen man während seines Studiums konfrontiert wird.

Um dieses Problem zu lösen gibt es z.B. den Quicksort, den Mergesort, den Bubblesort, den Swap-Sort und man könnte noch sehr viele weitere Algorithmen mit aufzählen, die dieses Problem lösen.

Es ist nun aber nicht immer sinnvoll diese Algorithmen direkt in den Klassen zu implementieren. Dies erhöht unweigerlich die Komplexität einer Klasse und erschwert die Wartung oder auch das Hinzufügen und Verändern der Algorithmen.

Wenn man diese Algorithmen nach den Regeln des Strategie-Pattern "outsourced", werden auch jegliche Bedingungsanweisungen hinfällig und unnötig. Man wählt zu Beginn eines der Strategien aus und verwendet es dann durchgehend.

Falls man die zu verwendende Strategie wechseln möchte, kann man dies tun, indem man die Anweisung, die man zu Beginn gesetzt hat, ganz einfach seinen Wünschen anpasst.

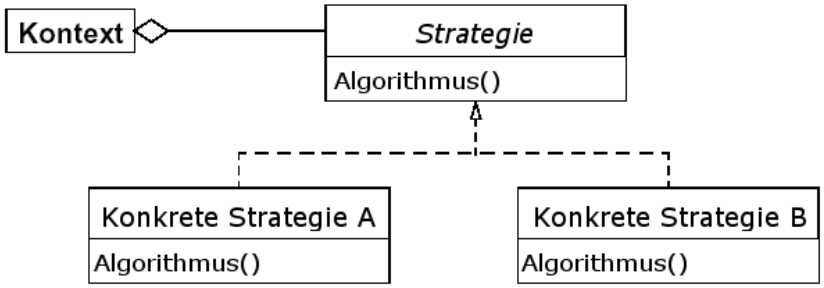

Abb. 6.4: Strategie-Pattern

Wenn man verschiedene verwandte Algorithmen hat, die sich nur im Verhalten unterscheiden, bietet sich hier das Verwenden des Strategie-Pattern an. Es ist so möglich verschiedene Algorithmen zur Lösung eines Problems zu implementieren und kinderleicht zwischen ihnen hin- und herzuwechseln.

Es stellt so eine Alternative zu den Unterklassen dar.

Dieses Pattern hat natürlich auch seine Nachteile.

Es steigt, da die Algorithmen nicht mehr direkt in den Klassen implementiert werden die diese Algorithmen benötigen, unumgänglich der Kommunikationsaufwand und oftmals auch die Anzahl der benötigten Objekte.

Außerdem muss die Klasse, die diese Algorithmen benötigt, die einzelnen Strategien kennen, damit sie zwischen ihnen auswählen kann. [9]

6.1.2 APDU

Die einfache Erstellung von CAPDUs und das einfache Parsen von RAPDUs ist eine wichtige Hilfsfunktionalität in unserer Arbeit.

Das Design ist einfach, es gibt jeweils eine abstrakte APDU Klasse und eine abstrakte RAPDU Klasse, die eine zum Erstellen, die andere zum Parsen. In diesen Klassen sind die Standardmethoden implementiert bzw definiert. So kann man in der APDU Klasse getAPDU() aufrufen um die APDU zu erhalten. Die APDU wird durch die abstrakte Methode generateAPDU() automatisch erstellt und zurückgegeben.

Dies allein ist nicht genug um APDUs zu erstellen, die verschiedenen APDU Typen müssen unterschiedlichen Anforderungen genügen, daher führt kein Weg an der manuellen Behandlung jedes APDU Typs vorbei, vorgenommen in der entsprechenden Klasse. Alle diese Klassen erben von der abstrakten Klasse, besitzen allerdings ihren eigenen Konstruktor und stellen die Setter für die benötigten Parameter und die generateApdu() Methode bereit.

Dabei sind die Methoden so ausgelegt, dass in der Benutzung möglichst wenig Information benötigt wird. In einigen Standardfällen oder bei einfachen APDUs kann man die APDU direkt nach Definition anfordern.

Falls man Secure Messaging verwenden möchte, verwendet man einfach anstatt der getAPDU() oder getResponse() Methoden die entsprechende secure Variante (getSecureAPDU() und getSecureResponse()). Wobei wichtig ist, dass jede CAPDU als secure APDU interpretiert werden kann, jede RAPDU auf eine secure APDU muss jedoch mit getSecureResponse() behandelt werden, sonst ist die Antwort fehlerhalt.

Zusätzlich wurde es vermieden, die CommandAPDU und ResponseAPDU Java Klassen zu benutzen, um die Klassen kompatibel zu älteren Java Versionen zu halten. Dies war zwar nicht gefordert, bat sich allerdings aufgrund der Struktur an, falls dies mal benötigt werden sollte. Dies bedeutet, dass anstatt dieser APDU Klassen für APDUs ausschließlich Byte Arrays verwendet werden.

Ein Beispiel für die Verwendung einer MSE Command APDU:

`MSEAPDU mse = new MSEAPDU()` //Der Konstruktor ist APDU spezifisch

`mse.setProtocol(mse.PACE_ID)` //Wir möchten diese MSE APDU für PACE benutzen

`mse.setOid(...)` //Hier wird ein Byte Array mit dem OID eingegeben, beispielsweise 0.4.0.127.0.7.2.2.4.2.2

`mse.setAuthenticator(new byte[] 0x03 )` //Wir möchten in PACE die PIN zur Authentifizierung benutzen, daher der Identifier 03. Es wird ein byte Array benötigt um diese Methode auch für andere APDUs zu benutzen.

`byte[] apdu = mse.getAPDU()` //Alternativ kann man auch die Methode `getSecureAPDU()` benutzen, für Secure Messaging.

Die MSE APDU ist ein etwas komplexerer Fall, wo viele Parameter gesetzt werden müssen, aber die Eingabe der benötigten Informationen können dem Benutzer nicht abgenommen werden.

Ein weiteres Beispiel, diesmal für das Parsen einer General Authenticate Response APDU:

`GeneralAuthRAPDU gaRapdu = new GeneralAuthRAPDU(rapdu)` //rapdu ist die empfangene rapdu in Byte Array form

`byte[] parsedResponse = gaRapdu.getResponse()` //Oder `getSecureResponse` wenn die RAPDU als Antwort auf eine secure Command APDU erfolgt.

Hierbei werden die Daten der rapdu zurückgegeben, wenn nur ein Identifier gefunden wurde. Der Header, die Identifier und alle Längenbytes werden abgeschnitten. Falls mehrere Identifier gefunden werden, bleiben die Identifier und Längenbytes erhalten, um die Response APDU Klassen nicht zu sehr aufzublähen. In einem solchen Fall müsste der Benutzer also selber die benötigten Teile der Daten extrahieren.

6.2 Server

Der Server erreicht nicht annähernd die Komplexität des Clients. Da, auf den Kommunikationsaufwand achtend, versucht wurde, soviele Aufgaben wie möglich in den Client unterzubringen, fällt dieser auch bedeutend umfangreicher aus.

Fast alle Berechnungen die der Server tätigt, müssen unumgänglich vom Server an den Client verschickt werden. So war es von großem Interesse, so wenig Berechnungen wie möglich im Server durchzuführen um die Kommunikation zwischen Client und Server zu reduzieren.

Im Server befinden sich also nur solche Aufgaben, die man nicht mit dem Client bewältigen konnte. Aufgaben, die, wenn man sie auf den Client verlagert hätte, z.B. Sicherheitskriterien verletzt hätten.

Wie beispielsweise bestimmte Teile des Terminal Authentication Protokolls. Bei der Erstellung der Signatur braucht man z.B. den privaten Schlüssel des Servers und da dieser den Server niemals verlassen sollte, muss die Signatur im Server erstellt (berechnet) werden.

So muss der Server die Signatur mit dem...

- RSA-Algorithmus [11][14]
- ECDSA [7]

Diese und ein paar weitere Funktionalitäten mussten wir also auf den Server verlagern.

Für die Erstellung von Signaturen bietet das Terminal Authentication Protokoll, wie schon erwähnt, den RSA- und den ECDSA-Algorithmus.

Da es aber möglich ist, dass sich dies in absehbarer Zukunft ändert, mussten wir hier auch wiederum eine einfache und möglichst schnelle Erweiterbarkeit gewährleisten können.

Aus diesem Grund wurde erneut das Strategie-Pattern angewendet. Dieses Mal gab es jedoch nur eine einzige Strategie-Familie, die Familien der Signatur Algorithmen.

6.3 eID-Protokoll

Die Aufgabe war es ein Protokoll zu entwickeln, welches die Ausführung von PACE, TA und CA ermöglicht. Die Protagonisten dieses Protokolls sind der Client und der Server, wobei der Client als Schnittstelle zwischen Kartenleser und Server dient, der Kartenleser, sowie der Server, also nur mit dem Client kommunizieren. Es mussten auch die Sicherheitskriterien beachtet werden, so dass einige Teile des Protokolls schon von vorne herein festgelegt waren.

Das Augenmerk bei der Entwicklung dieses Protokolls lag auf der Performanz. Es sollten so wenige Nachrichten wie möglich zwischen den einzelnen Protagonisten verschickt werden. Das Ziel war es also, soviele Aufgaben wie möglich auf dem Client ausführen zu lassen um den Kommunikationsaufwand so gering wie möglich zu halten.

Hätte man das Augenmerk auf ein anderes Ziel gelegt, wären noch andere Varianten des Protokolls möglich gewesen. Jedoch in Bezug auf unsere Ziele schien diese Variante des Protokolls als am Besten geeignet.

Wie schon oben erwähnt, müssen die Protagonisten mit diesem Protokoll in der Lage sein PACE, TA und CA auszuführen, im Folgenden sind nun die Überlegungen für ein eID-Protokoll zu jedem einzelnen Protokoll aufgeführt.

6.3.1 PACE

PACE dient dazu, basierend auf einem schwachen gemeinsamen Geheimnis, ein starkes gemeinsames Geheimnis zu generieren, mit dessen Hilfe man eine sichere Verbindung zwischen Client und Kartenleser aufbauen kann. Es reicht also aus, wenn PACE alleine zwischen Kartenleser und Client durchgeführt wird. Der Server muss an diesem Prozess nicht beteiligt werden. Der Kartenleser erhält zwar einen Public Key, der in Terminal Authentification wiederverwendet wird, dieser ist aber nicht ausschlaggebend für die Authentifizierung des Terminals.

Alle anderen Geheimnisse werden nur zwischen Client und Kartenleser benötigt und finden serverseitig keine weitere Verwendung.

6.3.2 Terminal Authentication

Bei der *Teminal Authentication*, authentifiziert sich der Terminal(der Server) bei der Karte. Es kann in diesem Schritt also nicht auf die Kommunikation mit dem Server verzichtet werden. Jedoch versucht man diese Kommunikation auf ein Minimum zu beschränken, da auf die Performanz ein Hauptaugenmerk während dieser Thesis gelegt werden soll.

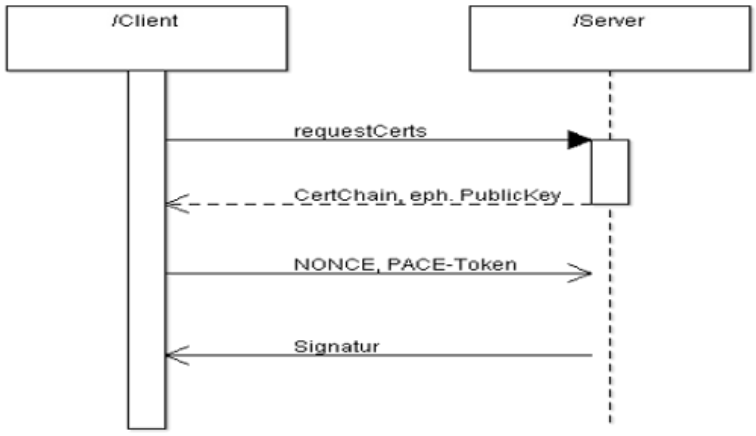

Abb. 6.5: *Nachrichtenaustausch des Protokolls bei Terminal Authentication*

Als erstes muss der Terminal der Karte eine gültige Zertifikatskette vorlegen (siehe oben). Die Karte verifiziert diese Kette und liest nach erfolgreicher Verifizierung den öffentlichen Schlüssel des Terminals aus. Um die Anzahl der Nachrichten, die zwischen Client und Terminal hin- und hergeschickt werden, zu

minimieren, schickt der Server alle Zertifikate auf einmal an den Client, welcher diese dann einzeln von der Karte verifizieren lässt.

Das Terminal erstellt ein vorübergehendes Diffie-Hellman Schlüsselpaar und übermittelt den öffentlichen Schlüssel dieses Paares, welcher später für die *Chip Authentication* benötigt wird, zusammen mit den Zertifikaten. Da die *Chip Authentication* dem Terminal dazu dient, die Authentizität der Karte festzustellen, muss dieses Schlüsselpaar im Terminal generiert werden und kann nicht auf den Client ausgelagert werden um die Kommunikation zu reduzieren.

Der Client erhält also in einer Nachricht die Zertifikatskette, sowie den öffentlichen Schlüssel des Terminals. Nachdem er alle Informationen und die Challenge der Karte übermittelt hat, erhält er eine Zufallszahl, ein sogenanntes Nonce, von der Karte. Er übermittelt dieses Nonce und das Token, welches man bei der erfolgreichen Ausführung von PACE erhalten hat, an das Terminal.

Um die Authentizität zu beweisen, erstellt das Terminal nun eine Nachricht, welche er mit seinem privaten Schlüssel signiert.

Im letzten Schritt schickt das Terminal die Signatur an den Client.

Für die *Terminal Authentication* sind also drei Nachrichten nötig. Aufgrund der Größe der ersten Nachricht, könnte es sein, dass man diese Nachricht unter Umständen noch einmal splitten muss. Die angewandte Strategie bietet jedoch ein Minimum an Kommunikation zwischen Client und Terminal.

6.3.3 Chip Authentication

Bei der *Chip Authentication* ist es möglich jegliche Kommunikation zwischen Karte und Client ablaufen zu lassen, wenn man den benötigten öffentlichen Schlüssel auf dem Client abspeichert. Dies widerspricht keiner der Sicherheitskriterien. Nachdem die Karte ihre Signatur an den Client geschickt hat, schickt dieser dem Server alle benötigten Informationen, damit dieser die Karte authentifizieren kann.

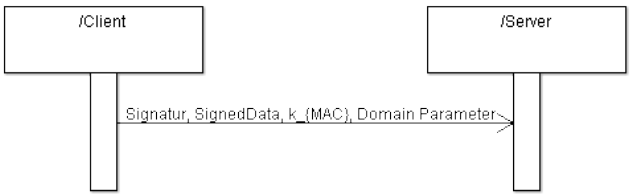

Abb. 6.6: *Nachrichtenaustausch des Protokolls bei Terminal Authentication*

Er schickt die Signatur der Karte zusammen ...

- ...mit der *SignedData*, welche den öffentlichen Schlüssel der Karte enthält
- ...dem Schlüssel, der benötigt wird, um die Signatur zu verifizieren
- ...und den Domainparametern ...

...an den Server, welcher nun mit den gegebenen Informationen die Authentifizierung der Karte durchführen kann.

6.3.4 Bewertung

Dieses Client-Server (CS1) Protokoll bietet einige Vorteil, aber auch gewisse Nachteile. Hier werden jetzt die Vor- und Nachteile des Designs diskutiert.

Gewiss ein Vorteil dieses Designs ist die Anzahl der, zu verschickenden Nachrichten. Es reduziert die Kommunikation auf ein Minimum.

Es gibt einen ähnlichen Ansatz (CS2), der das Protokoll mit dem gleichen Kommunikationsvolumen bewältigt, jedoch eine höhere Anzahl an verschickten Nachrichten aufweist. Anstatt einer gebündelten Nachricht bei der Chip Authentication, werden hier die Informationen in mehreren Nachrichten übermittelt.

Da dies den Zeitaufwand erhöht, setzte sich die CS1-Variante durch, die bei häufiger auftretenden Übertragungsfehlern jedoch gewisse Nachteile aufweist. Da einzelne Nachrichten größer sind, ist nämlich hier der zusätzliche Aufwand beim erneuten Versenden einer Nachricht höher.

Dieses Design erfordert fünf Nachrichten, die während des gesamten Protokolls hin- und hergeschickt werden (PACE: 0, TA: 4, CA: 1).

Während PACE werden zwei Geheimnisse erstellt. Das Erste ist nur vorübergehend, das Zweite wird in der darauffolgenden Terminal Authetication wiederverwendet. Beide Geheimnisse können im Client erstellt werden, da sie weder für die Authentifizierung des Terminals, geschweige denn für die Authentifizierung der Karte eine entscheidene Rolle spielen.

Es gibt insgesamt nur zwei Geheimnisse, die der Server für sich geheim behalten muss:

- seinen eigenen privaten Schlüssel
- den privaten Schlüssel des vorübergehenden Diffie-Hellman Schlüsselpaares, welches er während der Terminal Authentication selbst erstellt

Alle weiteren Geheimnisse lassen sich (wie oben beschrieben), ohne Verletzung der Sicherheitskriterien, auf dem Client erstellen und verwalten.

Kapitel 7

Fazit

Dieses Kapitel wurde von Clemens Deußer verfasst.

7.1 Ergebnis

In diesem Abschnitt behandeln wir die Frage, welche Ziele wir uns gestellt haben und ob diese erreicht wurden, sowie Probleme und Einschränkungen bei der Umsetzung.

Letztliches Ziel dieser Arbeit war die Implementierung einer eID Funktion auf Client-Server Basis, dies konnten wir leider nicht vollständig fertigstellen. Die gesetzten Meilensteine waren die initiale Kommunikation mit der Karte, die Umsetzung von PACE, danach den anderen Protokollen, die Möglichkeit auf die eID Funktion zugreifen zu können und schließlich die Adaption auf ein Client-Server System.

Alle Meilensteine bis auf den letzten wurden erfolgreich abgeschlossen, aufgrund von Zeitproblemen wird dieser letzte Schritt nicht von uns implementiert, wurde aber theoretisch in diesem Dokument behandelt.

Am meisten Zeit nahm mit Abstand die Implementierung von PACE in Anspruch. Hauptproblem war hierbei das Verstehen der Technischen Richtlinien des BSI. Wie wir abschließend feststellen konnten ist dieses Dokument größtenteils korrekt mit nur wenigen Fehlern, verstehen konnten wir dies jedoch aufgrund unserer mangelnden Erfahrung nur durch externe Hilfe. Zu der Verwirrung wurde durch eine Anfangs unvollständige und teilweise fehlerhafte Karte beigetragen, da die vollständig implementierten Karten noch nicht zur Verfügung standen. Eigene Fehler von nicht verschuldeten Fehlern zu trennen war somit sehr schwer und zeitraubend.

Ein weiteres Problem war die Fehlerfindung bei der Kommunikation mit dem Chip. Aufgrund der Sicherheitsarchitektur von PACE gibt der Chip bei Fehlschlagen eines der Protokollschritte am Ende des Protokolls eine generische Fehlermeldung zurück. Rückschlüsse auf den fehlgeschlagenen Schritt sind aus dieser Meldung nicht zu entnehmen, man muss also den Code bei der Fehlersuche komplett durchsuchen, was den Implementierungsaufwand erheblich erhöht.

Für den fertig implementierten ePA kann dies nicht anderes gemacht werden, da ansonsten die Sicherheit des Protokolls untergraben wäre, aber es wäre wünschenswert gewesen für die Implementierung Testkarten bereitzustellen, welche genauere Fehlermeldungen zurückgeben können. Welchen zusätzlichen Aufwand die Bereitstellung solcher Testkarten bedeutet hätte können wir schwer abschätzen, aber wir vermuten, dass es nicht trivial gewesen wäre.

Nach einiger Zeit bekamen wir verstärkte Unterstützung durch unseren Betreuer von FlexSecure, welcher über einige Erfahrung in diesem Gebiet verfügte und in direktem Kontakt zu der Bundesdruckerei stand. Somit konnten viele Fragen leicht beantwortet werden, was einige Probleme löste und einen erheblichen Fortschritt bedeutete.

Was wir in diesem Fall als Erfahrung mitnehmen konnten, ist sicherlich das strukturierte Vorgehen.

Man muss sich zunächst klar sein, was man eigentlich implementiert und man sollte dies in seiner Gänze auf einem niedrigen level gut verstanden haben. Falls dafür die vorhandene Dokumentation nicht ausreicht muss man sich selbst so viel zustätzliche Dokumentation besorgen wie man benötigt um die Protokolle in ihren Details und den Sinn dahinter zu verstehen. Sobald man den Prozess als Ganzes verstanden hat sind Fehlerquellen wesentlich leichter ausfindig zu machen und das Design der Implementierung wird stark vorangetrieben.

Man sollte allerdings nicht versuchen ein Dokument wie die technischen Richtlinien komplett durchzulesen, sondern stattdessen punktiert nach Informationen suchen, die man braucht um den nächsten Schritt zu verstehen. Dokumente dieser Art sind nicht aufeinander aufbauend strukturiert, sondern innerhalb des Dokumentes „verlinkt“. Das Dokument als Ganzes enthält somit alle Informationen, diese sind allerdings nicht immer zusammenhängend um ein exaktes, jedoch kompaktes Dokument zu erhalten.

Einen effizienten Weg zur Vereinfachung der bereits erwähnten Fehlersuche lieSS sich nicht finden, jedoch sollte man sicher nicht konventionelle Debugging Methoden verwenden. Man muss jede mögliche Fehlerquelle ausfindig machen und überprüfen, man muss also das Protokoll abstrakt abspielen und die möglichen Schwachstellen mit Wissen und Erfahrung erkennen, es könnte auch nur ein kleiner Flüchtigkeitsfehler sein (wie in unserem Falle ein Vorzeichenfehler). Ausgeprägtes Verständnis des Protokollablaufs auf der Seite der Karte ist hierbei von großer Hilfe.

Was uns persönlich nicht bewusst war, war die Kapazität und Robustheit des RFID Chips. Auch die Kommunikation mit APDUs fanden wir bemerkenswert. Über die Möglichkeit, größere Bitströme so herzurichten, dass die Bedeutung eindeutig ist, obwohl es letztlich „nur Einsen und Nullen“ sind, hatten wir uns zwar zuvor bereits Gedanken gemacht, aber ein solch durchstrukturiertes und in sich geschlossens Format wie APDUs zu studieren fanden wir sehr interessant.

Wie schnell und reibungslos dies zusammen mit der Übertragung funktioniert finden wir auch durchaus beachtlich, die Fortschritte in dieser Technologie sind hier nicht zu übersehen.

Unsere persönliche Gesinnung in Bezug auf dieses Projekt war zunächst nach Besprechung der Grundlagen beinahe euphorisch, aber mit der Zeit begannen wir uns mit dem Inhalt kritisch auseinanderzusetzen. Besonders die geplanten Eingriffe in die Anonymität, trotz aller guten Absichten, erschienen uns falsch und unfrei, aber dazu mehr in dem nächsten Abschnitt.

Wir hatten bereits zuvor an Projekten teilgenommen, die man nicht als freiheitsliebend bezeichnen kann, wie beispielsweise ein System, dass Arbeitnehmer komplett überwacht um automatisch Schlüsse über deren Verfügbarkeit zu ziehen. Aber dieses Projekt übertrifft in seiner Größenordnung alle bisherigen und unser „Bauchgefühl“ sagt uns, dass dies unter Umständen ein großer Schritt in die falsche Richtung sein könnte. Auch dazu mehr im nächsten Abschnitt, Auswirkungen auf unsere Arbeit hatte dies übrigens nicht.

7.2 Persönliche Meinung

Wenn ein Projekt wie dieses durchgeführt wird, kommt man an der Frage des Datenschutzes kaum vorbei. Fast die gesamte Kommunikation mit dem ePA dient letztlich dem Datenschutz und der Sicherheit, die Informationen selbst sind nur ein kleinerer Teil der Kommunikation.

Unsere persönliche Meinung, ob dies genug ist und ob es andere Probleme mit dem ePA gibt wird in diesem Kapitel besprochen.

Datenschutz in einem digitalen Zeitalter ist ein hochaktuelles Thema; einerseits geben große Teile der Bevölkerung freiwillig ihre Daten in Internetportalen wie Facebook, StudiVZ u.ä. preis, andererseits bedürfen persönlichen Daten gerade in einem digitalen Zeitalter besonderen Schutzes.

Der ePA wird diesem Datenschutz von einem kryptographischen Standpunkt aus gerecht. Szenarien, nach denen massenhaft Ausweise ohne Wissen des Inhabers (z.B. per Funk) ausgelesen werden können sind nicht gegeben und auch sonst hat man mit dem ePA vollständige Kontrolle über die enthaltenen Daten.

Der einzige Unterschied zum alten Ausweis besteht darin, dass er im Internet eingesetzt werden kann und soll. Es lohnt sich daher, sich etwas mit Daten im Internet zu beschäftigen.

Die grundlegenden Unterschiede zwischen „realen“ Daten (Papierdokumente u.ä.) und virtuellen Daten bestehen darin, dass virtuelle Daten mit minimalem Aufwand unendlich verbreitet werden können, keine Verfallsdauer besitzen und einfach durchsucht werden können.

In der Praxis bedeutet dies, dass alle Spuren, die man im Netz hinterlässt, potentiell für immer dort zu finden sind, ausserhalb der Kontrolle der betreffenden Person. Das allgemeine Mittel dagegen sind selbstgeschaffene Online Identitäten, also „falsche“ Namen und persönliche Daten, Anonymität.

Man stelle sich vor, diese Anonymität existiere nicht, jeder Benutzer im Internet müsste sich virtuell ausweisen können. Zunächst kein Unterschied zur realen Welt, aber aufgrund der Eigenschaften virtueller Daten ein großer Schritt zur Überwachbarkeit bzw Verfolgbarkeit.

Das anonyme Netz als Instanz gab es allerdings noch nie, letztlich kann alles zurückverfolgt werden. Die Schwierigkeit ist lediglich höher, ebenso wie Überwachung im realen Leben möglich aber ungleich schwerer als virtuelle Überwachung ist.

Hat der ePA als Ziel, dass jeder Bundesbürger sich online ausweisen muss? Nicht in absehbarer Zukunft, aber er stellt die Möglichkeit bereit.

Sind dies lediglich Bedenken einer kleinen Gruppe paranoider Datenschützer oder ist es ein reales gesellschaftliches Problem? Unserer Meinung nach steht die Gesellschaft diesem Problem nicht sensibilisiert und informiert gegenüber. Ein Beispiel ist eine Initiative, welche Unternehmen verpflichten würde, persönliche Daten nur dann weiterverbreiten zu dürfen, falls der Kunde dem explizit zustimmt. Derzeit muss er explizit ablehnen. Bei einer freien Wahl würde dies theoretisch die Menge ausschließen, denen es erklärtermaßen egal ist. Die Realität sieht allerdings anders aus.

Oft existiert die Möglichkeit zur Ablehnung nur abstrakt, beispielsweise in einem Brief an das Unternehmen, die Folge, dass die Daten weiterverbreitet werden, ist in Nutzungsbedingungen versteckt, welche kaum gelesen werden. Ein Benutzer lehnt somit oft nicht ab, weil er die Möglichkeit und die Konsequenzen nicht kennt. Ein unhaltbarer Zustand für jeden Datenschützer, der Antrag war berechtigt und erscheint jedem unabhängigen Betrachter als eine Korrektur einer Gesetzeslücke.

Falls jeder Kunde bei einer Registrierung einer Weiterverbreitung zustimmen müsste, so würde dies kaum einer tun, nur wenige Menschen mögen Werbung und Spam. Der Zusammenbruch einer Wirtschaft, deren Existenz den meisten Bürgern nicht einmal bewusst ist, stand bevor.

Die Initiative wurde fallengelassen, die entsprechenden Unternehmen hatten massive Lobbyarbeit geleistet. Arbeitsplätze waren gefährdet, von Leuten die mit Daten handeln damit jeder Bundesbürger zuverlässig seine tägliche Dosis Werbung im Briefkasten hat. Man könnte erwarten, dass solch ein Vorgang mehr Wellen erzeugt als ein verlorener Dienstwagen, aber die Reaktion war allenfalls verhalten, Werbung bekommt man eben, woher die Unternehmen wissen wo ich wohne? Steht doch in jedem Telefonbuch!

Bei aller Kritik glauben wir jedoch der Regierung, dass sie mit diesem Ausweis neue Möglichkeiten für die Wirtschaft schaffen und das Vertrauen der Bürger in das Netz stärken will, nur ein Punkt erschließt sich uns nicht:

Warum sind biometrische Daten vorgesehen? Diese Daten werden zunächst nur freiwillig sein, aber die Möglichkeit bereitzustellen ohne letztlich die Intention der flächendeckenden Einführung zu haben, ergibt keinen Sinn.

Wozu braucht man biometrische Daten? Die angeführten Gründe sind Fälschungssicherheit, Abwehr von Terrorismus und Verhinderung des Missbrauchs von gestohlenen Personalausweisen. Die EU hat zwar die biometrischen Daten bei dem ePass vorgeschrieben, jedoch existiert keine Verordnung dies auch bei dem ePA durchzuführen, ein solches Argument wäre somit nicht schlagkräftig.

Biometrische Merkmale machen Ausweise nicht fälschungssicherer, vor allem nicht den digitalen. Es ist davon auszugehen, dass eine Person, welche diesen

Ausweis inklusive aller Protokolle und Zertifikate fälschen kann, auch die zugehörigen biometrischen Daten fälschen kann. Der Fälschungssicherheitsgewinn ist aus kryptographischer Sicht gleich null. Der neue Ausweis ist noch sicherer als der Alte, und bereits von dem alten Ausweis wurden weniger als 0.001% Fälschungen bekannt (Quelle: Anfrage an die Bundesregierung nach Anzahl der gefälschten Ausweise. Insgesamt 495 Fälschungen von 62 Millionen ausgestellten Exemplaren in den Jahren 2001 bis 2007, entspricht einer Quote von etwa 0.0008%) und ist (war) somit eines der sichersten Dokumente weltweit.

Außerdem ist kein Akt des Terrorismus bekannt, wo ein gefälschter Pass eingesetzt wurde, auch ist die Intention des Terrorismus nicht an einem Fingerabdruck erkennbar. Der einzige Gewinn im Kampf gegen den Terrorismus wäre ein Abgleich des Fingerabdrucks mit einer Datenbank, aber diese Prozedur ist nach Angabe der Bundesregierung nicht vorgesehen, die Fingerabdrücke werden nicht einmal zentral gespeichert und selbst falls jemand im Innenministerium solche Hoffnungen hegt, die gefährlichste Art Terrorist ist vermutlich nicht vorbestraft und somit wäre der Fingerabdruck nicht registriert.

Dem Missbrauch von gestohlenen Personalausweisen, von Leuten die einem ähnlich sehen, wäre mit einem Fingerabdruck zu beheben, aber wie real ist eine solche Gefahr? Fälle dieser Art sind wenig bis gar nicht bekannt. Eine einfache Überprüfung der Daten würde in den meisten Fällen den Missbrauch bereits beweisen, sich eine gesamte Identität zu schaffen ist wesentlich schwerer als einen Ausweis zu stehlen. Der Sicherheitsgewinn wäre zwar vorhanden aber sehr minimal und sicherlich kann er kein Ausschlag für die flächendeckende Einführung von biometrischen Ausweisen sein.

Der Fingerabdruck auf dem Ausweis kann nur drei Ziele haben:

- Er setzt den Zusammenhang zwischen Identität und Individuum unwiderruflich fest (symbolisch),
- Mit diesem Zusammenhang können kriminalistische Ermittlungen erheblich erleichtert werden und
- Authentifizierungen können erleichtert werden, falls man den Fingerabdruck als Passwortersatz benutzt.

Fingerabdrücke als Bequemlichkeit? Durchaus nicht irreal, die eindeutige und einfache Identifizierung per Fingerabdruck wäre definitiv zukunftsfähig, aber dafür ist der ePA eigentlich gar nicht ausgelegt.

Die eigenen Daten als schützenswert zu erkennen, weil dies mit der Souveränität des Individuums verbunden ist, erfordert ein Problembewusstsein, welches sich nicht aufbaut weil der gesamte Vorgang ausserhalb der unmittelbaren Wahrnehmung stattfindet.

Aufgabe des Staates wäre es in diesem Falle, seine Bürger gerade deshalb trotzdem zu schützen und die entsprechenden Gesetze und Bestimmungen an ein neues Zeitalter anzupassen. Vielleicht ist ein Online Telefonbuch ein Eingriff in die Privatsphäre, obwohl dort nichts anderes steht als in der alten Buchversion? Vielleicht sollte man, bevor man die Möglichkeit einer sicheren Authentifizierung bereitstellt, zunächst die Verwendung solcher Daten einschränken?

Die derzeitige Situation erscheint uns wie ein riesiges Experiment, bei dem keiner so richtig weiss wohin es führt. Werden die zunehmenden Möglichkeiten der Überwachung irgendwann wirklich repressiv genutzt werden? Wird die Privatsphäre wirklich dem gläsernen Menschen weichen, dessen Interessen, Freunde, soziale Zugehörigkeit und Wohnort auf Knopfdruck abrufbar sind?

Zunächst ist dies nichts weiter als ein Horrorszenario, aber ohne ein solches werden die Gefahren der ersten Schritte nicht deutlich, daher sei uns die Erwähnung erlaubt.

Vereinfachungen bei Identifikationen finden wir ebenso wünschenswert wie einen sensibleren Umgang mit persönlicheren Daten. In die Zukunft schauen können wir nicht und wir sind uns der Tatsache bewusst, dass wir aufgrund unserer Spezialisierung stärker sensibilisiert sind, aber wir haben den Eindruck, dass die Kompetenz in den relevanten Stellen mit Interessenvertretern der falschen Sorte besetzt ist und das ist bedenklich.

Kapitel 8

Ausblick

In diesem Kapitel werden Punkte besprochen, welche in dieser Arbeit nicht diskutiert wurden, jedoch in dem Zusammenhang interessant sind. Ausserdem werden die fortschreitenden Möglichkeiten dieser Arbeit diskutiert, was die nächsten Schritte unserer Implementierung sind, was noch benötigt wird für ein funktionsfähiges System und welche Einsatzbereiche sich bieten würden.

Dieses Kapitel wurde von Clemens Deußer verfasst.

8.1 Nicht behandelte Gebiete

8.1.1 Sicherheitsanalyse

Die Frage nach einer Sicherheitsanalyse haben wir bisher nicht gestellt, sie wäre allerdings durchaus interessant.

Das Passwortsystem des ePAs erklärt sich wie bereits erwähnt wie folgt:

- Sowohl die CAN als auch die PIN kann benutzt werden um Daten in dem Ausweis auszulesen
- PIN ist geheim, CAN ist nicht geheim
- Man sollte sich im Besitz des Ausweises befinden um die CAN auszulesen

Soweit die Theorie, der letzte Punkt bedarf einer Klarstellung: Man kann die CAN auch kennen ohne in Besitz des Ausweises zu sein, es ist schließlich eine statische Nummer. Sobald man einmal im Besitz des Ausweises war, so kennt man die CAN und kann sie für diesen Ausweis beliebig verwenden. Es gibt zwar das Modell einer dynamischen CAN, welche durch ein Display auf dem Ausweis angezeigt werden kann, dies wird allerdings von uns nicht benutzt.

Sowohl CAN als auch PIN können für jegliche Auslesung der Daten verwendet werden, ob Online oder aus der Entfernung oder bei der Polizeikontrolle, Einschränkungen gibt es zunächst nicht. Warum also die beiden Passwörter? Bei Identifizierung mit CAN kann man belegen, dass man sich im Besitz des Ausweises befindet, was allerdings überflüssig ist, da man bei Identifizierung durch einen Ausweis ohnehin in dessen Besitz ist. Die PIN macht mehr Sinn,

da dadurch nicht nur der Besitz des Ausweises, sondern auch die rechtmässige Identität des Inhabers geprüft wird, nur er kennt die PIN.

Der Sinn erschließt sich somit leicht, der Ausweis soll immer leicht auslesbar sein, aber massenhafte Profilerzeugung soll verhindert werden. Gleichzeitig soll der Ausweis zur authentifizierten Identifizierung verwendet werden können, wozu der einfache Besitz des Ausweises nicht ausreicht.

Ist dies ein sicheres System oder basiert es letztlich darauf, dass eine umfassende Profilerstellung durch CANs nicht realisierbar ist? Falls man die CAN einer bestimmten Person kennt, was durchaus passieren kann, schließlich ist die CAN nicht geheim, und die Möglichkeit hat Ausweise in einem bestimmten Umkreis auszulesen. so könnte man somit nachprüfen ob sich diese Person innerhalb dieses Umkreises befindet. Ob dies ein reelles Sicherheitsrisiko ist, ist eine andere Frage, aber prinzipiell besteht diese Möglichkeit, auch weil die CAN nicht ausreichend gegen Brute Force Angriffe geschützt ist.

Eine interessante Problematik, aber zu weit ausserhalb des Rahmens dieses Dokuments für eine genauere Betrachtung.

Auch haben wir uns nicht mit der generellen Sicherheitsanalyse über ein einfaches Verständnis hinaus beschäftigt. So wird Diffie-Hellman verwendet, obwohl dies bereits angreifbar ist, aber dafür mit einem Passwort abgesichert. Das Protokoll wird so komplexer und widersetzt sich der einfachen „Man in the middle“ Attacke. Ob dies aber bedeutet, dass keine anderen Attacken existieren oder wie widerstandsfähig dieses System ist, wäre Gegenstand einer solchen Sicherheitsanalyse.

Muss ein System wie der ePA überhaupt über ähnlich sichere Mechanismen verfügen wie ein größer angelegtes Sicherheitssystem? Die Inhalte unterliegen zwar dem Datenschutz aber der finanzielle/wirtschaftliche Wert dieser Informationen ist gering. Zusätzlich kann man den Ausweis schlecht über längere Zeiträume belauschen, da er sich nicht im Besitz des Angreifers befindet und mobil ist, die Ausgangslage ist also eine schlechtere für einen Angreifer im Vergleich zu einer statischen Situation beispielsweise über das Internet.

Abschließend ist zwar der Eindruck vorhanden, dass der ePA im Sinne der Spezifikation durchaus sicher ist, aber eine ausführliche und umfassende Sicherheitsanalyse benötigt viel Erfahrung, hohe mathematische sowie kryptographische Kompetenz und viel Zeit. Daher passte sie nicht in den Rahmen dieser Ausarbeitung.

8.1.2 Brisanz der eGK

Die eGK (elektronische Gesundheitskarte) wurde zwar in unserer Arbeit beschrieben, aber weiter sind wir nicht auf sie eingegangen. Sie bedarf besonderer Erwähnung, da sie, im Unterschied zu anderen eCards, ähnlich wie der ePA verpflichtend ist für Mitglieder der entsprechenden Krankenkassen.

Die eGK wird noch kontroverser diskutiert als der ePA, da sie nicht nur zur Identifizierung der Person, sondern auch der Krankheit und allem was damit in Verbindung steht, dient. Es sind zwar keine direkten Informationen zu der Krankheit enthalten, aber man kann dies wohl aus Rezeptdaten ableiten. Eine Meinung ist, dass man dadurch die Kontrolle über die eigene Behandlung entzogen bekommt. Während man aktuell nur 10 Euro Praxisgebühr zusätzlich

für eine zweite Meinung bezahlen muss, indem man die erste Behandlung verschweigt, so wäre das in der eGK gespeichert, man müsste also einen Grund vorbringen.

Genau dies, also die Verhinderung von Missbrauch und unnötigen Untersuchungen, ist allerdings ein erklärtes Ziel dieser eGK und man kann die Position der Krankenkassen durchaus verstehen.

Man kann sich diesen Konflikt durch zwei Beispiele verdeutlichen:

Auf der einen Seite steht der Patient, der alles besser weiss als der Arzt und sich die gleiche Meinung gleich fünfmal einholt und dadurch beträchtliche zusätzliche Kosten verursacht. Im Interesse des Gesundheitssystems sollte dieses Verhalten entweder unterbunden werden oder der Patient soll die Koste selber tragen.

Auf der anderen Seite steht ein Patient, der von einem schlechten Arzt behandelt wurde und sich dagegen schwerlich wehren kann. Dieses Beispiel ist in Deutschland noch sehr realitätsfern, in anderen Ländern ist es allerdings ein Problem dem sich viele Patienten ausgesetzt sehen.

Man kann dies sogar länderübergreifend beobachten. In Schweden wird den Patienten die Wahl der Behandlung und des behandelnden Arztes größtenteils entzogen. Die Behandlung ist zwar kostenlos, aber wenn man durch Pech einen schlechten Arzt erwischt kann man sich kaum dagegen wehren.

Auf der anderen Seite steht Amerika, dessen Einwohner in nahezu paranoider Art jegliche mögliche Einmischung in ihre Rechte als Patienten strikt ablehnen. Die Folge ist das teuerste Gesundheitssystem der Welt, mit der schlechtesten sozialen Versorgung unter den westlichen Industrienationen.

Ich bezweifle jedoch, dass das schwedische Modell als befürchtetes Ziel Grund der Protestbewegung ist, vielmehr sehen Patienten ihren Gesundheitszustand als privat an und somit die Kontrolle durch ein Lese/Schreibgerät als Eingriff in ihre Privatsphäre.

Das deutsche Gesundheitssystem scheint sich in die schwedische Richtung zu bewegen, mit der einfachen Begründung, dass es sich sonst nicht finanzieren lässt. Eine genauere Beurteilung der Situation und ausführliche Auseinandersetzung mit beiden Seiten wäre ein interessantes Thema was allerdings nicht im Fokus unserer Arbeit stand.

8.2 Weiterführende Implementierung

Wie bereits im Fazit erwähnt, haben wir es nicht geschafft die ursprünglich geplante Client-Server Architektur für dieses Projekt umzusetzen. Dies wäre somit der nächste Schritt in diesem Projekt, das Programm müsste in die entsprechenden Pakete aufgeteilt und angepasst werden.

Obwohl wir bereits den Großteil einer provisorischen UI implementiert haben, müsste man dies durch eine robuste, ergonomische und schönere Version ersetzen, welche den intuitiven Umgang mit dem Programm erlaubt.

Sobald dies fertiggestellt wäre, müsste das Programm eine QS (Qualitätssicherung) Phase durchlaufen, mit zahlreichen Tests und Fehlerkorrekturen. Sobald diese Phase abgeschlossen ist und die Version 1.0 zur Verfügung steht, könnte man das Programm einsetzen.

Die möglichen Einsatzgebiete sind schwer einzugrenzen. Aufgrund der spezialisierten Natur der Applikation wäre sie vermutlich schwer im eGovernment einsetzbar, dort würde man vermutlich auf die eCard-API zurückgreifen. Warenanbieter könnten ein derartiges Programm für den Internetversand allerdings benötigen und für Altersüberprüfung wäre es ebenfalls gut geeignet. Die einzige Hürde wäre in diesem Fall die Notwendigkeit eines Kartenlesers, ein nicht zu unterschätzendes Problem.

Literaturverzeichnis

[1] Chipkarten-Strategie der Bundesregierung. www.einblick.dgb.de/hintergrund/2008/13/chipkartenstrategie.pdf.

[2] Gemeinsame Pressemitteilung des Bundesministerium für Wirtschaft und Arbeit, des Bundesministeriums des Innern, des Bundesministeriums für Gesundheit und Soziales und des Bundesministeriums der Finanzen, März 2005. http://www.teletrust.org/uploads/media/2005-pm-e-card.pdf.

[3] ISO/IEC 10116:2006. Information technology - Security techniques - Modes of operation for an n-bit block cipher, 2006.

[4] ISO/IEC 7816-8:2004. Identification cards Ű Integrated circuit cards Ű Part 8: Commands for security operations, 2004.

[5] BSI. Technische Richtlinie TR-03110, Version 2.01. https://www.bsi.bund.de/cae/servlet/contentblob/532066/publicationFile/27971/TR-03110_v201_pdf.pdf.

[6] BSI. Das eCard API Framework, März 2005. https://www.bsi.bund.de/cln_136/ContentBSI/Publikationen/TechnischeRichtlinien/tr03112/index_htm.html.

[7] BSI. Elliptic Curve Cryptography (ECC) Version 1.11, 2009.

[8] Sommerakademie des Unabhängigen Landeszentrums für Datenschutz Schleswig-Holstein. eCard Strategie der Bundesregierung, August 2005. https://www.datenschutzzentrum.de/sommerakademie/2005/somak05_engel.pdf.

[9] Ralph Johnson John Vlissides Erich Gamma, Richard Helm. Entwurfsmuster - Elemente wiederverwendbarer objektorietierter Software, 2004.

[10] Russel Housley. Cryptographic message syntax (CMS), 2004.

[11] Jakob Jonsson and Burt Kaliski. Public-key cryptography standards (PKCS)#1: RSA cryptography specifications version 2.1, 2003.

[12] Bernd Kowalski. eCard Strategie der Bundesregierung, Februar 2007. http://www.dmkn.de/itsecurity.nsf/D2C9995A91A43B80C12572E300525ECD/\protect\T1\textdollarFile/ecard_strategie_der_bundesregierung_vortrag_kowalski_bsi.pdf.

[13] Bernd Kowalski. eCard API Framework in Deutschland und Europa, November 2008. http://www.ptb.de/de/aktuelles/archiv/presseinfos/pi2008/pitext/ecard_api_vortrag.pdf.

[14] RSA Laboratories. PKCS#1 v2.1: RSA cryptography standard, 2002.

[15] Microsoft. Kurze Einführung in ASN.1 und BER. http://support.microsoft.com/kb/252648/de.

[16] NIST. Data Encryption Standard (DES), 1999.

[17] NIST. Specification for the Advanced Encryption Standard (AES), 2001.

[18] NIST. Secure hash standard (and Change Notice to include SHA-224), 2002.

[19] NIST. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, 2005.

Abbildungsverzeichnis

Tabellenverzeichnis