Evaluierung von Chipkartensystemen für HUELKA

Studienarbeit von
Erik-Oliver Blaß

1. April 2000

Europäisches Institut für Systemsicherheit, Universität Karlsruhe

Betreuer:
Prof. Dr. Thomas Beth

Betreuende Mitarbeiter:
Dr. Willi Geiselmann
Dipl. Inform. Harald Lukhaub

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Zusammenfassung

Im Rahmen des Projektes HUELKA[1] werden Buchungen von einem Bearbeiter innerhalb seines Web-Browser eingegeben und zu einem zentralen Server übertragen. Um Authentizität und Fälschungssicherheit der Buchung zu gewährleisten, sollen neben der herkömmlichen Verschlüsselung der Daten während ihres Transports auch digitale Signaturen mit Hilfe von Chipkartensystemen eingesetzt werden.

In dieser Arbeit erkläre ich zunächst die Anforderungen an die Chipkartensysteme im Projekt HUELKA, die Anforderungen, die sich aus dem Gesetz zur digitalen Signatur ergeben und die wichtigsten bzw. bekanntesten Typen und Standards für moderne Chipkartenlesegeräte und Chipkarten: CT-API, PC/SC, PKCS#11, OpenCard, sowie ISO-7816 (Dateisystem), Windows for Smartcard, MULTOS und JavaCard. Passend zu diesen Standards folgt eine Übersicht über Leistung und Preis von Chipkarten und Lesegeräten von diversen Herstellern am Markt, die für HUELKA in Frage kommen, sowie eine Bewertung. Abschließend beschreibe ich einige Erfahrungen, die ich mit dem Entwicklungskit Cyberflex Access SDK der Firma Schlumberger gesammelt habe.

Eine Anbindung der Chipkarten und Lesegeräte mit „OpenCard“ erscheint am sinnvollsten, da OpenCard komplett auf Java basiert und damit einfach aus dem Applet der HUELKA-Kryptoumgebung heraus angesprochen werden kann.

Als Lesegerät bietet sich das Chipdrive Micro der Firma Towitoko an, da es sehr klein, preiswert, einfach seriell anschließbar und OpenCard kompatibel ist. Eine JavaCard fähige Chipkarte eignet sich aus ähnlichen Gründen: sie ist mit wenig Aufwand OpenCard kompatibel und außerdem im Funktionsumfang nicht eingeschränkt - da relativ einfach programmierbar und somit stark erweiterbar.

Ein Problem stellt das strenge Gesetz zur digitalen Signatur dar: es fordert - neben bestimmten Algorithmen auf der Chipkarte - die generelle Anforderung ITSEC „E4“ an die Chipkartenhardware, sowie „E2“ an Lesegeräte. Keine OpenCard fähige Chipkarte erfüllt derzeit diese Anforderung. Ein Public-Key Zertifikat muß darüberhinaus von einer bestimmten Zertifizierungsstelle erstellt werden, damit eine digitale Signatur gesetzeskonform ist. Allerdings fehlt HUELKA ein richtiges TrustCenter für eine gesetzeskonforme digitale Signatur: das Rechenzentrum der Universität Karlsruhe ist keine registrierte Zertifizierungsstelle.

[1] Haushalts Überwachungsliste Karlsruhe

Kapitel 1

Einleitung

1.1 HUELKA

Im Juli 1998 hat der Ministerrat des Landes Baden-Württemberg die Einführung eines Globalhaushalts im Rahmen des „Führungs- und Informationssystems (Controlling)“ beschlossen. Als ersten Schritt müssen die Hochschulen Baden-Württembergs mit einer Kosten- und Leistungsrechnung ab Januar 2000 beginnen [14]. Die Universität Karlsruhe führt daher zum 1. Januar 2000 einen globalisierten Haushalt und damit verbunden die Kosten- und Leistungsrechnung ein. Im Rahmen des Projektes „HUELKA“[1] soll diese Kosten- und Leistungsrechnung mit Hilfe des Computers unterstützt werden.

In HUELKA soll die Kontoführung eines Instituts, also z.B. das Tätigen von Buchungen, rechnergestützt und - in Zukunft - komplett papierlos auf einem zentralen Server durchgeführt werden, s. Abbildung 1.1. Der Bearbeiter gibt an einem Client-Computer seine Buchung innerhalb eines Browser-Fensters in ein Java-Applet ein, das daraufhin die Aufgabe übernimmt, die Daten zwecks Weiterverarbeitung zu dem zentralen Server zu transportieren.

An diese Datenübertragung werden die klassischen kryptographische Anforderungen [22] gestellt:

1. Geheimhaltung,
2. Integrität,
3. Verfälschungssicherheit,
4. Authentizität.

[1] Haushalts Überwachungsliste Karlsruhe

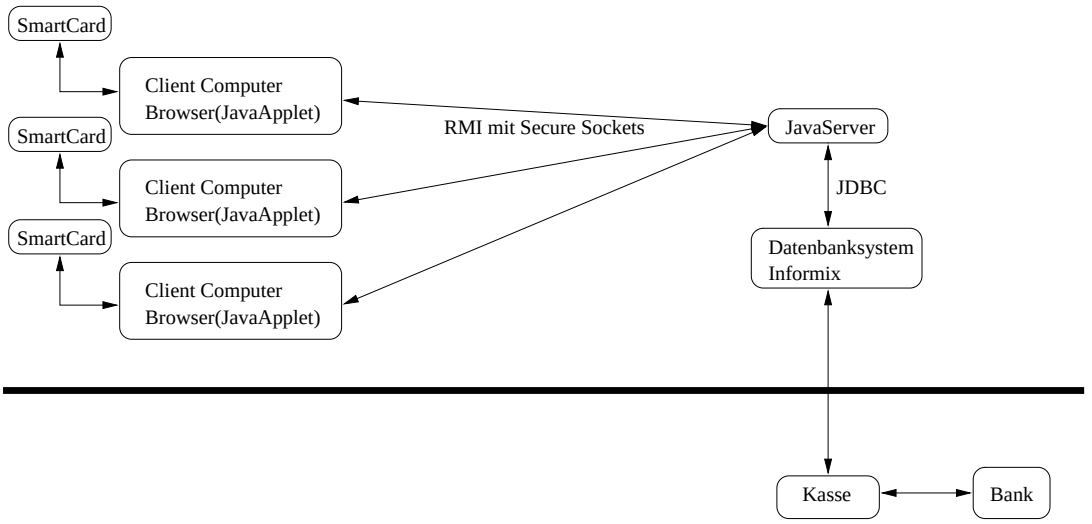

Abbildung 1.1: Projekt HUELKA

Während die ersten drei Anforderungen innerhalb des HUELKA Java-Applet implementiert worden sind, z.B. mit RMI[2] über „Secure Sockets“ (ähnlich SSL[3]), soll die Authentizität mit der Hilfe von Chipkartensystemen erreicht werden.

1.2 Authentizität durch digitale Unterschrift

Üblicherweise werden klassische „Papier“-Buchungen - z.B. auf den bekannten Formblättern - durch eine handschriftliche Unterschrift des Ausstellers authentisch: die Unterschrift sichert dem Empfänger, der in der Lage ist, diese Unterschrift zu überprüfen, zu, daß die Buchung authentisch - also echt - ist. Das heißt in diesem Falle, daß die Buchung von einer Person mit der entsprechenden Berechtigung ausgestellt worden ist.

Auch elektronische Buchungen können unterschrieben werden. Das elektronische Äquivalent zu einer handschriftlichen Unterschrift unter einem Papierdokument wird üblicherweise mit „digitaler Unterschrift“ oder „digitaler Signatur“ bezeichnet:
Von einem elektronischen Dokument (z.B. einer Buchung) wird ein Hashwert erzeugt und dieser dann „signiert“, d.h. mit einem geeigneten Signieralgorithmus (z.B. RSA) unterschrieben. Das Ergebnis dieses Vorgangs ist die digitale Unterschrift [8].

Der Empfänger von elektronischem Dokument und digitaler Unterschrift kann

[2]Remote Method Invocation, [19]
[3]Secure Sockets Layer, [10]

mit Hilfe einer „Zertifizierungsstelle“[4] Dokument und Unterschrift auf Echtheit überprüfen.

In HUELKA soll die Authentizität der elektronischen Buchungen mit Hilfe einer digitalen Signatur gewährleistet werden. Dies geht nicht nur mittels reiner Softwaremethoden (wie z.B. PGP), sondern auch mit Hilfe moderner Chipkartensysteme[5]. Chipkartensysteme haben gegenüber Software einige Vorteile, z.B. sicherheitstechnischer oder beweistechnischer Art, die sich im weiteren selbst erklären.

1.3 Digitale Unterschrift mit Chipkartensystemen in HUELKA

Im folgenden werden die Anforderungen und das Szenario beschrieben, innerhalb dessen die Chipkarten und Lesergeräte in HUELKA eingesetzt werden sollen:

An das Computersystem des Benutzer (üblicherweise ein Intel PC mit Betriebssystem Windows NT) ist ein Chipkartenlesegerät - welcher Form auch immer - mit einer dazu passenden „krypto fähigen“ Chipkarte angeschlossen. „Krypto fähig“ bedeutet hier, daß die Karte dazu imstande ist, kryptographische Operationen wie Verschlüsseln, Entschlüsseln oder Signieren durchzuführen.

Das Buchungsprogramm (ein Java-Applet) läuft in einer Java 1.2 fähigen Virtual Machine (z.B. Sun Java 1.2 Plugin für Netscape oder Internet Explorer). Soll eine Buchung abgeschickt werden, so berechnet das Applet von der Buchung einen Hashwert und übergibt ihn der Chipkarte, die zu dem Hashwert eine Signatur anfertigt und zurückliefert. Diese Signatur wird mitsamt der Buchung dann an den zentralen Server geschickt.

Zum Erstellen einer Unterschrift muß die Karte einen entsprechenden Signier/Public-Key Algorithmus beherrschen. Darüber hinaus sollte der private Schlüssel auf der Karte erzeugt und gespeichert werden können, so daß er aus Sicherheitsgründen nie ausgelesen werden kann und die Karte niemals verläßt. Einer eventuellen Attacke zum Auslesen oder Erraten des geheimen Schlüssels sollte die Karte vorbeugen. Sinnvoll aus Gründen der Offenheit gegenüber späteren Erweiterungen ist es zudem, daß das Zertifikat, d.h. der vom Trust Center unterschriebene öffentliche Schlüssel mit zusätzlichen Informationen über den Inhaber der Karte ebenso auf der Karte speicherbar und exportierbar ist. Mit Hilfe des Zertifikats kann der Server die Unterschrift zu dem Dokument überprüfen.

Die Signiervorgänge sollten konform mit dem Deutschen „Gesetz zur digitalen

[4] auch mit Trust Center oder CA - Certificate Authority bezeichnet

[5] Chipkartensystem: die Kombination von Chipkartenlesegerät und (Microprozessor-) Chipkarte

Signatur“ (siehe Abschnitt 2.1) sein, da sie dann im Gegensatz zu einfachen elektronischen Daten einen gewissen juristischen Beweiswert besitzen.

Da das Buchungsprogramm ein Java-Applet ist, sollte die Anbindung von Chipkarte und Lesegerät aus einem Applet heraus möglichst einfach zu realisieren sein.

Wünschenswert wäre ebenso die Möglichkeiten zur einfachen Erweiterung der Chipkarte, beispielsweise das Programmieren eigener Karten-Applikationen, denn damit könnte man z.B. Studentenausweise und Mensa-Chipkeys durch eine einzige Chipkarte ersetzen.

Kapitel 2

Signaturgesetz

In HUELKA sollen digital erzeugte Unterschriften möglichst konform mit dem „Gesetz zur digitalen Signatur“ sein, da diese Konformität u.U. einen gewissen juristischen Beweiswert mit sich bringt.

Was ist das „Gesetz zur digitalen Signatur“, was bedeutet dieser „Beweiswert“? Welche Anforderungen stellt das Gesetz an ein Chipkartensystem, so daß damit angefertigte digitale Signaturen dem Signaturgesetz entsprechen und sie so ähnlich den bekannten handschriftlichen Unterschriften einen gesteigerten juristischen Beweiswert besitzen?

2.1 Problemstellung

Papierdokumente mit handschriftlicher Unterschrift gelten vor Gericht als Urkunde und haben - im Falle einer Streitigkeit - nach §§416, 440 Zivilprozeßordnung so „erhöhten Beweiswert“: Sie geben „sofern sie von den Ausstellern unterschrieben [...] sind, vollen Beweis dafür, daß die in ihm enthaltenen Erklärungen von den Ausstellern abgegeben sind.“ [3].

Elektronische Dokumente hingegen sind leicht zu ändern: z.B. den Inhalt einer Email oder ihre Absenderadresse zu fälschen ist - auch ohne daß der Betrug auffällt - sehr einfach möglich. Daher haben elektronische Dokumente vor Gericht keinen bzw. einen sehr geringen Beweiswert [3].

Die Kryptographie bietet jedoch die Möglichkeit mit Hilfe von digitalen Unterschriften (siehe Abschnitt 1.2) Authentizität auch für elektronische Dokumente zuzusichern. Um für digitale Signaturen eine Rechtsbasis zu schaffen, hat der Gesetzgeber das sogenannte „Signaturgesetz“ [1] und dazugehörende Verordnungen verabschiedet, die genau festlegen, was eine digitale Signatur ist, d.h. unter welchen Umständen eine wie geleistete digitale Unterschrift juristisch als signaturgesetz-konform gelten soll.

Das Signaturgesetz beschreibt - neben Anforderungen an die Zertifizierungsstellen, auf die hier nicht eingegangen wird - die Eigenschaften, die die technischen Komponenten, in unserem Falle also die Chipkarten und Lesergeräte, erfüllen müssen, um rechtsgültige Unterschriftenn zu erzeugen. Im folgenden werden diese Eigenschaften beschrieben.

2.2 Geeignete Algorithmen

Die zum Signieren verwendeten Algorithmen sind nach [8] genau bestimmt. Nur die Benutzung folgender Algorithmen zum Hashen und Signieren eines elektronischen Dokumentes entsprechen dem Signaturgesetz:

1. Hashfunktionen
    - RIPE-MD 160
    - SHA-1
2. Signieralgorithmen
    - RSA
    - DSA
    - EC-DSA
    - EC-GDSA
    - EC-NR
    - EC-KCDSA

Bei der Wahl der in [8] angegebenen Parameter bezüglich Schlüssellänge sind die Algorithmen bis 2004 sicher.

Da z.B. für die Erzeugung der Schlüsselpaare Zufallszahlen benötigt werden, schreibt das Signaturgesetz auch hier bestimmten Anforderungen vor: zur Erzeugung von Schlüsselpaaren soll ein Algorithmus möglichst basierend auf einem physikalischen Zufallszahlen-Generator verwendet werden. Für andere Aufgaben - z.B. zum eigentlichen Signieren - kann ein Pseudozufallszahlen-Generator benutzt werden. Der Bundesanzeiger [8] verweist hier auf eine entsprechende IEEE-Norm [20].

2.3 Schlüssel-Zertifikate

Auf den Chipkarten sollen exportierbare und von einer Zertifizierungsstelle unterschriebene Zertifikate gespeichert werden. Ein solches Zertifikat auf der Chipkarte) muß folgende Informationen enthalten [7]:

- Name des Signaturschlüsselinhabers,
- öffentliche Signaturschlüssel,
- Bezeichnung der Algorithmen zum Benutzen der Schlüssel,
- laufende Nummer des Zertifikats,
- Beginn und Ende der Gültigkeit des Zertifikats,
- Name der Zertifizierungsstelle und,
- Angaben, ob die Benutzung des Schlüssel eingeschränkt ist.

Diese Zertifikate dürfen nur von bestimmten, von der Regulierungsbehörde für Post und Telekommunikation registrierten Zertifizierungsstellen ausgestellt werden, damit sie nach Signaturgesetz gültig sind. Bisher sind nur die Deutsche Telekom, und zur CeBit 2000 auch die Deutsche Post AG registrierte Zertifizierungsstellen. Ein Zertifikat ist für maximal fünf Jahre gültig.

2.4 ITSEC

Neben obigen Voraussetzungen sind im weiteren die Eigenschaften der ITSEC[1] Evaluierungsstufen aus Tabelle 2.1 zu erfüllen, siehe hierzu [17].

Alle technischen Komponenten - dazu gehört auch Signiersoftware und Zufallszahlengenerator - müssen von entsprechenden „Zertifizierungsstellen für technische Komponenten“[2] nach der entsprechenden ITSEC Norm zertifiziert werden, für Chipkarten ist dies E4, für Lesegeräte gilt E2.

Die derzeit nach Signaturgesetz zertifizierten technischen Komponenten sind in Tabelle 2.2 abgebildet.

2.5 Beweiswert der digitalen Signatur

Im Signaturgesetz fehlt die Aussage, was genau die Konformität für die Beweiskraft einer digitalen Signatur bedeutet. Als Beweisstück vor Gericht gilt ein handschriftlich unterschriebenes Papier Dokument als „Urkunde“, d.h. in einem Streitfalle ob der Authentizität des Dokumentes hat das Gericht in seiner Urteilsfindung keinerlei Spielraum bei der Gewichtung dieses Beweises: der Beweis ist als (wahre) Tatsache hinzunehmen.

[1] Information Technology Security Evaluation Criteria

[2] zur Zeit sind das debis, BSI und TÜV Informationstechnik

Aufgabe	Stufe
Erzeugen von Signaturschlüsseln	E4
Speichern und Anwenden des privaten Schlüssels	E4
Technische Komponenten zur geschäftsmäßigen Nutzung durch dritte	E4
Erfassen, Speichern und Anwenden der Identifikationsdaten	E2
Hashen der zu signierenden Daten	E2
Erkennbarmachen des Inhalts zu signierender Daten	E2
Prüfen einer digitalen Signatur	E2
Überprüfen von Zertifikaten	E2
Vergabe eines Zeitstempels	E2

Tabelle 2.1: Evaluierungsstufen für technische Komponenten, aus [6]

Name der Komponente	Hersteller
Trust Center Schlüsselgenerator	Telekom
Funktionsbibliothek TCrypt-TCM	Telekom
Telesec Signaturkarte	Telekom
Trust Center Zertifikateserver	Telekom
CardMan Chipkartenlesegerät	Utimaco
SafeGuard Verschlüsselungssoftware	Utimaco

Tabelle 2.2: Zertifizierte technische Komponenten nach [5]

Da digitale Dokumente keine Urkunden sind [21] und das Signaturgesetz darüber hinaus keinerlei Aussage über den Beweiswert einer gesetzeskonformen digitalen Unterschrift macht, gelten elektronische Dokumente auch signiert nur als „Augenscheinsbeweis“ und unterliegen so einer freien Beweiswürdigung (einem hohen Ermessungsspielraum) nach §286 ZPO. Damit hat ein digital signiertes Dokument einen geringeren und unsichereren Beweiswert als ein handschriftlich unterschriebenes Papierdokument.

Diese zur Zeit unsichere Beweislage der digitalen Signatur verbunden mit den hohen Kosten für eine Zertifizierung sind ein Grund daür, daß bisher - außer der Telekom und Utimaco - kein Hersteller seine Produkte hat zertifizieren lassen.

Der Nachteil der unsicheren Beweislage soll mit einer umfassenden Änderung der entsprechenden Gesetzestexte im Sommer 2000 beseitigt werden [2]. Ein signaturgesetz-konform unterschriebenes elektronisches Dokument soll einer Urkunde damit gleichgestellt werden. Eine Änderung des Europarechts bewirkt außerdem die europaweite Einführung der Beweiskraft digitaler Signaturen mit ähnlichen Kriterien.

Kapitel 3

Kartenstandards

Die Geschichte der Chipkarte beginnt 1950, als „Diner's Club“ ein kleines bedrucktes Stück PVC in der bekannten Kartenform zum bahrgeldlosen Bezahlen nur „mit dem guten Namen“ anbietet [13]. VISA, Mastercard und alle anderen Kreditkartenunternehmen ziehen später nach. Da die bedruckten Karten schwer maschinenlesbar und relativ leicht zu fälschen sind, folgen kurz darauf Karten mit Magnetstreifen. Auf dem Magnetstreifen sind die Informationen gespeichert, die vorher auf die Karte gedruckt worden sind. Einen Schritt weiter gehen die Deutschen Dethloff und Grötrupp. Sie lassen 1968 die ersten ICCs[1] patentieren [13]. In diese Karten ist ein kleiner Chip eingesetzt, daher der Name „Chipkarte“.

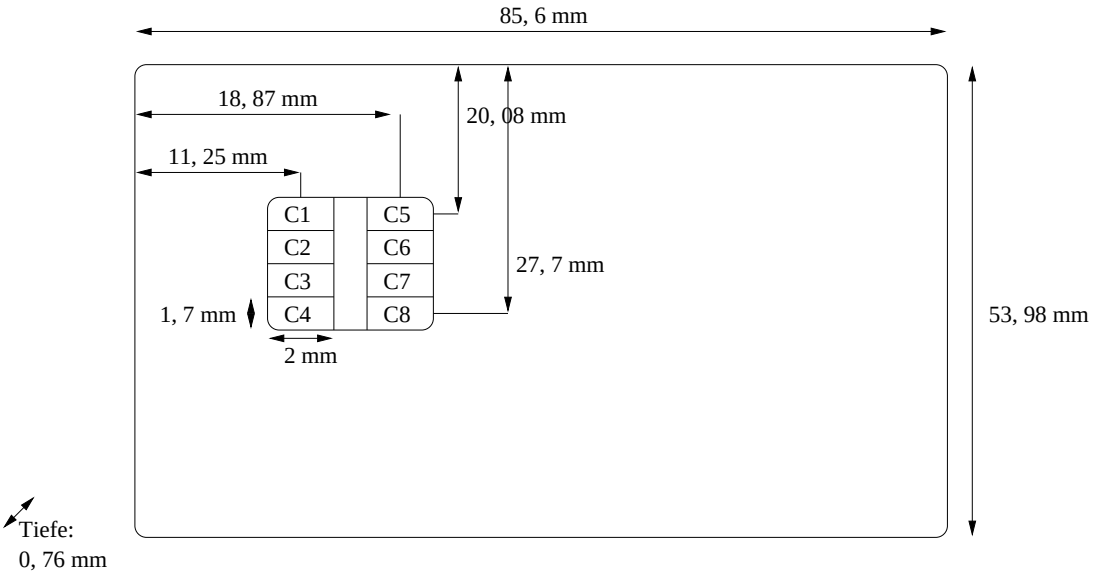

Abbildung 3.1: Layout nach ISO-7810, ID-1

[1] Integrated Circuit Cards, im Englischen auch mit „Smartcard“ bezeichnet

Der Chip erlaubt zunächst nur das Hineinschreiben und das Auslesen von Daten über einige Kontaktpunkte auf dem Chip. Neben diesen reinen „Speicherkarten“ mit ROM oder EEPROM (heutige Anwendung: Telefonkarten und Krankenkassenkarten) entstehen bald richtige „Microprozessorkarten“. Dies sind komplette Minicomputer mit Mikroprozessoren, CPU (8 bit, ca. 5 MHz), RAM (ca. 1 KByte), ROM (ca. 24 KByte) und EEPROM (ca. 16 KByte), einem Betriebssytem, Dateisystem und mehreren Applikationen, die auf der Chipkarte ausgeführt werden [13]. Einige Karten besitzen neben der CPU auch Coprozessoren, wie z.B. einen kryptographischem Coprozessor zum Verschlüsseln von Daten oder Berechnen von digitalen Signaturen.

Die ISO Norm 7810 beschreibt die physikalischen Eigenschaften von verschiedenen Kartentypen wie Dimension, Robustheit gegenüber Verformung, Temperaturbeständigkeit usw. [13]. Der Kartentyp ISO 7810 ID-1 aus Abbildung 3.1 hat das uns bekannte „Kreditkartenformat“.

Neben den Kontaktkarten - die Karten, die zu einem Lesegerät durch Kontaktstellen auf dem Chip in Verbindung treten - existieren auch kontaktlose Karten, die per Induktionsprinzip Daten übetragen. Diese Lösungen sind allerdings recht teuer, fehleranfällig und für eine Umgebung, in der sehr häufig die kryptographische Funktion der Karte benötigt wird, eher ungeeignet. Es gibt auch spektakuläre Formen von Smartcards, wie z.B. der Java IButton[2], der die Gößе und Form eines Knopfes hat.

Die Chipkarte, bzw. die Applikation, die auf der Chipkarte ausgeführt wird, soll mit der Applikation auf dem PC kommunizieren. Diese Kommunikation wird von einer „darunterliegenden Schicht“ folgendermaßen realisiert [9]:

- die PC-Applikation schickt Daten an den Chipkartenleser,
- der Chipkartenleser schickt die Daten weiter an die Chipkarte, die sie verarbeitet,

und umgekehrt

- die Chipkarte schickt eine Antwort an den Chipkartenleser,
- der Chipkartenleser schickt diese Daten weiter an die PC-Applikation.

Dies bedeutet für die Entwicklung einer eigenen Chipkartenapplikation, daß nach drei Standard-Schnittstellen implementiert werden muß, und zwar für die Kommunikation zwischen:

1. PC und Lesegerät. Dies ist beschrieben in Abschnitt 4.

[2]http://www.ibutton.com

2. Lesegerät und Chipkarte. Bei Chipkarte und Lesergerät funktioniert der Datenaustausch komplett über den Anschluss C7 (s. Abbildung 3.1). Die dafür zuständigen Protokolle (z.B. T=0, T=1, ...) sind nach ISO 7816-3 festgelegt und werden an dieser Stelle nicht weiter beschrieben (siehe [13]). Die modernen Chipkarten sind meistens vom Typ T=0 oder T=1, und die Leser unterstützen sowohl T=0 als auch T=1. Die „Verständigung" Leser - Karte stellt also kein Problem für die Entwicklung einer Applikation dar.

3. PC-Applikation (z.B. dem Applet im Browser des Benutzer) und den Funktionen, die die Chipkarte, bzw. dessen Betriebssystem, zur Verfügung stellt (z.B. Dateisystem-, oder kryptographische Funktionen). Auf die Standards der Chipkarten wird im Folgenden eingegangen.

Um die eventuellen kryptographischen Fähigkeiten der Karte zu benutzen, das Dateisystem anzusprechen und neue Applikationen für die Karte zu entwickeln, muß das Kartenbetriebssystem, bzw. dessen API[3] angesprochen werden.

Neben proprietären Lösungen für Bestriebssysteme und Chipkarten-APIs haben sich folgende Standards etabliert:

3.1 ISO 7816-4 (Filesystem Karten)

Die weitverbreitete Norm ISO 7816-4 ist ein Betriebssystemstandard für Chipkarten. Sie legt einen Befehlssatz, ein Dateisystem und das Datenaustauschformat für Mikroprozessor-Karten fest.

Der ISO 7816-4 Befehlssatz umfaßt die unterschiedlichen Operationen zur Manipulation des Dateisystems und einige kryptographische Befehle.

Das Dateisystem auf der Chipkarte ist der bekannten hierarchischen Baumstruktur des DOS-Dateisystem ähnlich. Es gibt Unterverzeichnisse („DF - Dedicated Files"), sowie einfache Daten- oder Programmdateien („EF - Elementary Files"). Jedes Unterverzeichniss wiederum kann sowohl weitere DFs als auch EFs enthalten. Darüberhinaus wird das „Wurzelverzeichnis" mit MF („Masterfile") bezeichnet. Der Anwender kann per Befehl Dateien anlegen, löschen und verändern oder bestimmte Programme auf der Karte starten [28].

Datenpakete, d.h. Befehle und Parameter vom Rechner an die Karte und umgekehrt, werden in sogenannte APDUs[4] „eingepackt" und an die Chipkarte gesendet, siehe Abbildung 3.2.

[3] Application Program Interface
[4] Application Protocol Data Unit

Instruktions Typ 1 Byte	Instruktions Code 1 Byte	Parameter1 1 Byte	Parameter2 1 Byte	Breite Datenfeld 1 Byte	Datenfeld	Breite Antwortfeld 1 Byte

Abbildung 3.2: APDU Format für den Datenaustausch

3.2 JavaCard

Soll für eine Chipkarte eine eigene Applikation entwickelt werden, so muß der Programmierer in einer bestimmten Sprache (u.U. in Assembler) ein kartenspezifisches Programm schreiben: er schreibt sein Programm speziell für die Funktionen die das API der Karte bereitstellt. Der Nachteil liegt auf der Hand: bei einem Wechsel auf einen anderen Kartentyp - oder in einer Umgebung mit verschiedenen Karten - muß jeweils wieder ein neues Programm speziell für die neue Karte geschrieben werden. U.U. sind alle Programme auf der Karte auch festverdrahtet, d.h. sie wurden beim Erzeugen bzw. „Personalisieren“ der Karte „festgebrannt“. Dann können keine weiteren Applikationen auf die Karte gespielt werden.

Die Idee hinter der „JavaCard“ [23] ist es, eine Java Virtual Machine auf der Chipkarte zur Verfügung zu stellen. Ein Anwender, der seine eigene Applikation für die Chipkarte entwickeln will (entsprechend „Applets“ werden die Applikationen hier „Cardlets“[5] genannt), kann diese so komplett in Java implementieren. Darüberhinaus sollen diese „Cardlets“ auf verschiedenen JavaCards ablauffähig sein, d.h. die eigene Kartensoftware muß bei einem Wechsel der Karten nicht neu programmiert werden. Der Programmierer kann seine Cardlets in das Dateisystem der Karte zu jedem Zeitpunkt herunterladen und dann per Befehl starten, löschen usw. Die JavaCard Virtual Machine ist „striped-down“, d.h. sie enthält nicht den vollen Umfang einer Java 1.2 Virtual Machine: so fehlt z.B. die Unterstützung für mehrere Threads, es gibt kein dynamisches Nachladen von Klassen und einige Datentypen, wie mehrdimenionale Arrays oder große Datentypen wie double, long, ... [26] fehlen.

JavaCard bietet dem Cardlet Programmierer ein API mit diversen Kryptofunktionen an, wie Hash-, Signier- und Verschlüsselungsalgorithmen[6], die er für seine Cardlets nutzen kann.

Das neue JavaCard 2.1 ermöglicht darüberhinaus den Datenaustausch zwischen zwei Cardlets, sowie das Erzeugen von Schlüsseln auf der Karte.

Der eigentliche Datenaustausch zwischen Cardlet und PC - Applikation, bzw. dessen Middleware (s. Abschnitt 4) erfolgt über APDUs.

[5] Cardlet ist abgeleitet aus (Smart) Card und Applet
[6] SHA-1, MD-5, RIPE MD-160, RSA, DSA, (3)-DES

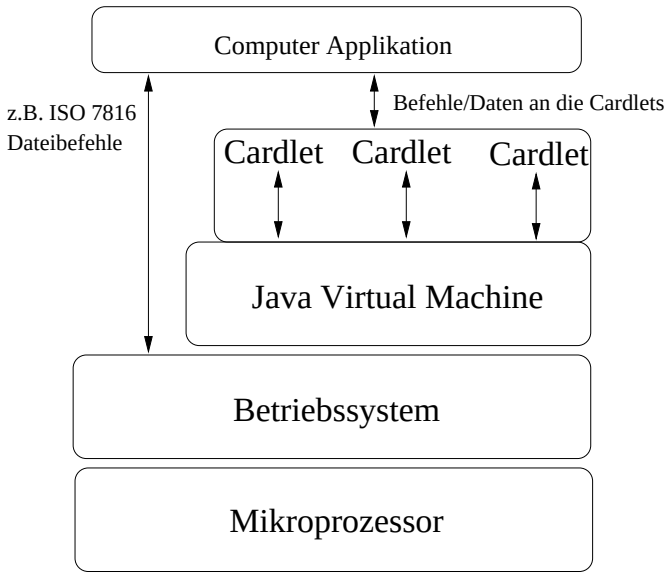

Abbildung 3.3: Architektur der JavaCard, aus [26]

Eine Erweiterung von JavaCard ist Visa Open Platform. VOP legt unter anderem fest, wie Cardlets vom Computer auf die Karte gespielt und dort verwaltet werden, siehe [30].

JavaCard Virtual Machines untereinander sind nur Source-Code kompatibel - und (noch) nicht bytecode-binär kompatibel, wie die klassischen Virtual Machines.

3.3 Windows for SmartCard

Ähnlich den JavaCards verfolgt „Windows for Smartcard“ den Ansatz, die Arbeit für Programmierer von Chipkarten-Applikationen zu erleichtern. Mit Microsoft Visual Basic und Visual C++ können Applikationen chipkartenunabhängig entwickelt und auf die Karte geladen werden. Das Windows Betriebssystem für die Karte ist ausserdem in Komponenten unterteilt, parametrisier- und anpassbar an die Wünsche des Benutzers. So sind z.B. die Eigenschaften wie die Größe des Dateisystems, die benutzbaren kryptographischen Protokolle und verschiedene Befehle der Virtual Machine einzustellen [18]. Auf die Karte werden also nicht nur die Applikationen gespielt, sondern auch die für die Applikationen benötigten Komponenten des Windows Betriebssystems.

3.4 MULTOS

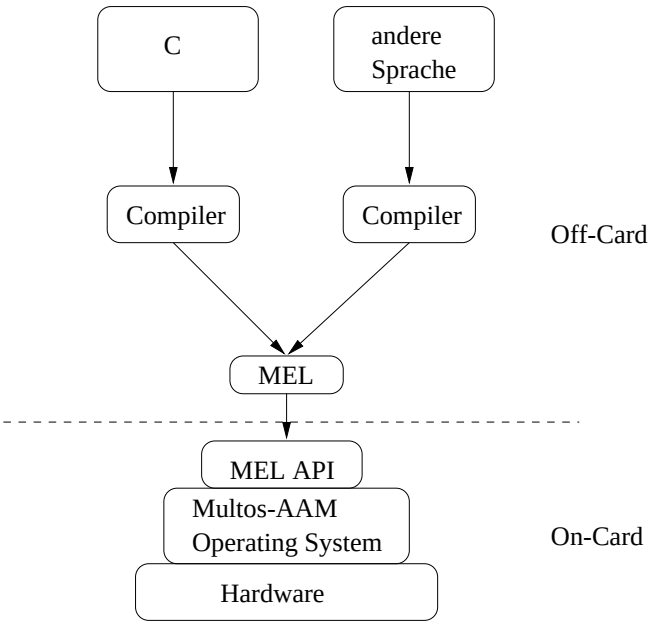

Abbildung 3.4: MULTOS, aus [11]

Auch das Kartenbetriebssystem MULTOS[7] ermöglicht das dynamische Speichern und Starten von selbst entwickelten Karten-Applikationen. Neben einem ISO 7816 Dateisystem bietet MULTOS über die Programmiersprache MEL[8] die Grundlage zur Entwicklung von eigenen sogenannten MEL-„Codelets“. C-Programmierer - oder Programmierer anderer Programmiersprachen - können ihre Anwendungen nach MEL konvertieren [11]. Die so entstehenden MEL-Codelets können auf jeder beliebigen MULTOS-Karte geladen und ausgeführt werden. Dabei sorgt auf der Chipkarte die Application Abstract Machine (AAM) - ähnlich der Java Virtual Machine - für eine Umsetzung der MEL-Befehle in die Befehle des Chipkartenbetriebssystems (s. dazu Abbildung 3.4). Jeder Entwickler muß für jede seiner Applikationen bei „Moasco“, dem Konsortium, das MULTOS pflegt, eine weltweit eindeutige 2 Byte lange ID beantragen. Es kann also maximal 65535 verschiedene MULTOS Applikationen geben [11].

[7] Multi-Application Operating System for Smart Cards
[8] Multos Enabling Language

Kapitel 4

Middleware

Um den Zugriff auf die Funktionalität der Chipkartensysteme für Applikationen zu vereinfachen, existieren wiederum API Standards. Es geht bei dieser „Middleware“ meist um mehr, als um reine Treiberschnittstellen für verschiedene Hardware (Leser und Chipkarten), sondern auch um abstrakte Funktionalitäten wie „Ressource Manager“ und „Service Provider“. Sinn dieser Middleware ist es, daß Anbieter von Chipkartensystemen entsprechende Low-Level Funktionen ihrer Produkte implementieren und in die entsprechenden Stellen im Middleware API einfügen. Der Programmierer einer Applikation kann dann das (einfache) Hochsprachen-API der Middleware programmieren und muß sich um die internen Vorgänge im Leser und auf der Chipkarte keine Gedanken machen. Ändert sich das Chipkartensystem - werden andere Karten oder Leser benutzt, so braucht die Applikation dafür nicht neu programmiert zu werden.

Die Middleware arbeitet traditionell nach dem Entwurfsmuster „Einzelstück“ [24]: Beim Inititalisieren des Middlewaresystems werden die herstellerspezifischen Implementierungen in einen globalen Zugriffspunkt geladen und stehen ab dann unter diesem Zugriffspunkt zur Verfügung.

4.1 CT-API

Der Standard CT-API[1] wurde 1996 von der Deutschen Telekom, der GMD, dem TÜV Informationstechnik und TeleTrust Deutschland definiert. Er ist eine Beschreibung nur für Schnittstellen, die Lesegeräte zur Verfügung stellen sollen und bietet keine weitergehende Abstraktion von Chipkartenfunktionalität, wie PC/SC oder OpenCard an. CT-API ist recht einfach aufgebaut, seine Schnittstelle besteht nur aus drei Befehlen, siehe Tabelle 4.1.

Anhand des Parameters „dad“ wird bei CT_data erkannt, ob das Kommando

[1] Card Terminal Application Programming Interface

CT-API Funktion	Bedeutung
CT_init (ctn, pn)	CT-API und die Verbindung Computer - Lesergerät Nr. ctn mit Slot pn initialisieren
CT_data (ctn, dad, [...], command, [...], response)	Datenaustausch mit Leser Nr. ctn an Adresse dad, Befehl Command, Puffer für Antwort in response
CT_close (ctn)	Verbindung zu Leser Nr. ctn beenden

Tabelle 4.1: CT-API 1.1 (mit den wichtigsten Parametern), aus [16]

ein Befehl für das Lesergerät selbst oder ein Datenpaket (APDU) für die Chipkarte ist.

Zu den Lesegerät-Befehlen, die CT-API anbietet[2], gehören Befehle wie RESET, GET STATUS, EJECT ICC, oder INPUT und OUTPUT für Geräte mit Display und Tastatur.

CT-API abstrahiert keine Chipkartenfunktionalität, es kann Befehle und Daten (z.B. in Form von APDUs) durch den Leser nur an die Karten weiterreichen.

4.2 PC/SC

Microsoft, Gemplus SA, Bull, Hewlett-Packard, IBM, Schlumberger, Siemens Nixdorf, Sun Microsystems, Toshiba, VeriFone und Intel legen seit 1996 die „Interoperability Specification for ICCs and Personal Computer Systems“ fest [31]. Ziel dieser PC/SC[3] Workgroup ist es, die Zusammenarbeit von Microsoft Windows PCs mit Chipkartensystemen zu vereinfachen.

In Abbildung 4.1 ist der „schichtenartige“ Aufbau des PC/SC Systems zu erkennen.

Ein PC/SC System besteht neben Chipkarten (ICCs) aus Interface Device Handlern[4], einem Ressource Manager und mehreren Service Providern.

Die Aufgabe eines Interface Device Handlers ist es, verschiedene Lesegeräte und deren Anschlußmöglichkeiten (USB, seriell, ...) hinter einer Schnittstelle zu verstecken. Die Schnittstelle bietet als Funktionalität die Kommunikation mit der Chipkarte, sowie deren Zustands-Überwachung (z.B. Chipkarte eingelegt, herausgenommen, ...).

[2] CT-BCS, Card Terminal Basic Command Set, [27]
[3] PersonalComputer SmartCard
[4] Interface Device (IFD) - Lesegerät

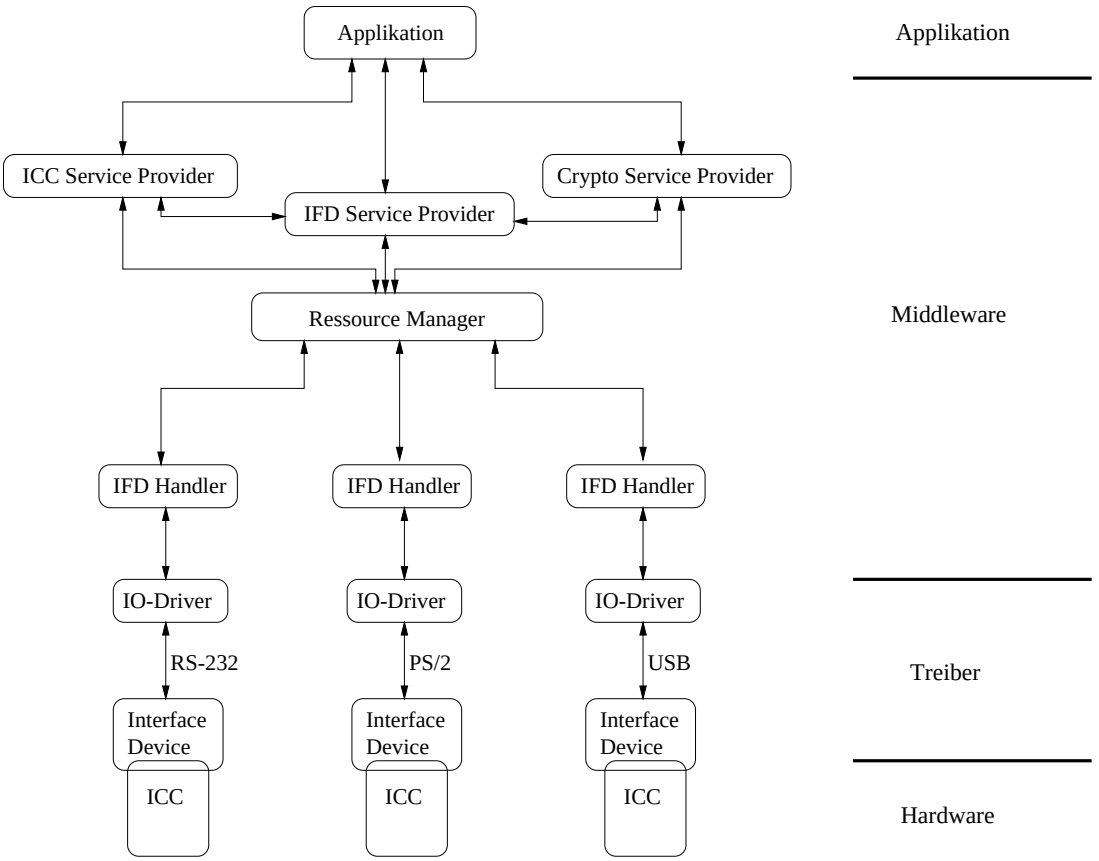

Abbildung 4.1: PC/SC Aufbau nach [32]

Der Ressource Manager löst drei Probleme beim Zugriff auf IFDs und deren Chipkarten [31]:

1. Zustandsüberwachung über die dem Computer angeschlossenen IFDs, sowie deren Chipkarten. Dazu gehört die Bereitstellung und Weitergabe dieser Informationen an beim Ressource Manager registrierte Applikationen.

2. Der Ressource Manager kontrolliert den Zugriff auf Karten und Lesegeräte. Dadurch wird ermöglicht, daß mehrere Applikationen gleichzeitig auf Funktionen von einem Lesegerät und einer Chipkarte zugreifen oder aber eine Applikation den Zugriff „exklusiv“ erhält.

3. Bereitstellung von Kommunikationsprimitiven für „darüberliegende“ Schichten: komplexe Funktionen kann die Chipkarte nur in mehreren aufeinanderfolgenden Schritten ausführen, deren Steuerung wird vom Ressource Manager übernommen.

Die Service Provider (ICC Service Provider und Cryptographic Service Provider) abstrahieren die implementierungstechnischen Details verschiedener Chipkarten und bieten deren Funktionalität über eine Schnittstelle an. Der ICC Service Provider bietet Dateisystemfunktionalität und Authentifizierungsprimitive (z.B. PIN) an. Den US-Exportbeschränkungen unterliegen die Cryptographic Services: sie bieten Schlüsselverwaltung inkl. Erzeugung, Hash- und Signierfunktionen, sowie Verschlüsselung. Um die „extended capabilities“ [32] der Lesegeräte, wie z.B. Display und PIN-Pad kümmert sich der IFD Service Provider.

Dem Applikations-Programmierer stellt das API für jede dieser „Schichten“ entsprechende Bibliotheksfunktionen zur Verfügung. Bisher extistiert nur eine offizielle Implementierung der PC/SC API Bibliotheken, und zwar eine Microsoft Version für Intel Win32 Systeme. Weitere Plattformen (interessant wären z.B. Windows CE oder NT auf Alpha/Mips) sind bis heute auch nicht geplant. Im Rahmen des MUSCLE [4] Projekts wird eine „inoffizielle“ PC/SC Bibliothek für Linux implementiert.

4.3 OpenCard

Die Liste der Mitglieder des OpenCard Consortiums ist lang: 3-G International, American Express Travel Related Services, Bull, Dallas Semiconductor, First Access, Gemplus, IBM, Toshiba Corporation, Towitoko, Schlumberger, Siemens, Sun Microsystems, UbiQ Inc., Visa International, sowie XAC Automation. Im Gegensatz zum PC/SC Ansatz - für eine Plattform (Windows), ein API zu realisieren, ist die Idee des OpenCard Frameworks die, möglichst plattformunabhängig zu ein: das OpenCard Framework[5] ist ein plattformunabhängiges, rein in Java geschriebenes API nur für Java. Damit ist OpenCard programmierbar und benutzbar auf jedem Computer, für den es eine Java Virtual Machine gibt - vom Network Computer über den PC bis hin zum Großrechner.

Das OpenCard Framework selbst besteht aus zwei Teilen [12]:

1. Card Service Layer
   Dies ist eine Schnitstelle zum Zugriff auf die Funktionalität der Karte, bzw. den Applikationen auf der Karte. Dazu gehören die FileAccessCardServices, für den Umgang mit Dateien und Dateisystemen der Karte, sowie die SignatureCardServices für kryptographische Funktionen wie Signieren und Verschlüsseln. Teil der Card Services ist auch das Application Management: dies verwaltet die Applikationen auf der Chipkarte. Es bietet Funktionen wie die Installation einer Applikation, Auflisten, Starten und Blockieren von Applikationen.

[5] oder kurz: OCF

2. Terminal Service Layer
   Dieser Teil bietet den Zugang zu den physikalischen „Slots“ der Lesegeräte: die Abfrage des Zustands der Karte, das Aushandeln von Kommunikationsparametern. Er erlaubt das Senden und Empfangen von APDUs. Bei entsprechenden Geräten können Dienste wie PIN-Pad oder Daumenabdruck-Leser angesprochen werden.

Die Funktionsweise von OpenCard ist vergleichbar mit der von PC/SC.

Chipkartenhersteller bieten für ihre spezielle Chipkarte eine Implementierung des Card Service Layers an, genauso wie Lesegerätehersteller das Card Terminal Layer implementieren.

Das OpenCard Framework wird in Suns Solaris Version 8 enthalten sein.

4.4 PKCS #11

Die Firma RSA definiert mit PKCS #11 den „Cryptographic Token Interface Standard“, auch „Cryptoki“ genannt. Cryptoki ist eine Schnittstelle zwischen Applikation und den sog. „cryptographic tokens“. Ein Token ist ein Gerät, das Kryptographie „kann“. Es unterstützt kryptographische Funktionen wie Ver- und Entschlüsseln, Verifizieren und Signieren [15] und speichert kryptographische Objekte, wie Schlüsselpaare oder Zertifikate. Token können z.B. Chipkarten oder andere Hardwaresysteme sein - es existieren aber auch reine „Software-Tokens[6]“.

Ein „Slot“ aus Abbildung 4.2 ist ein physikalischer Zugangspunkt für ein Token, so z.B. ein Leser für Chipkarten.

Netscapes Browser „Communicator“ kann PKCS #11 Tokens einbinden und zur Ausführung kryptographischer Operationen ansteueren.

[6] http://www.trustcenter.de/html/ Produkte/TC_PKCS11/1490.htm

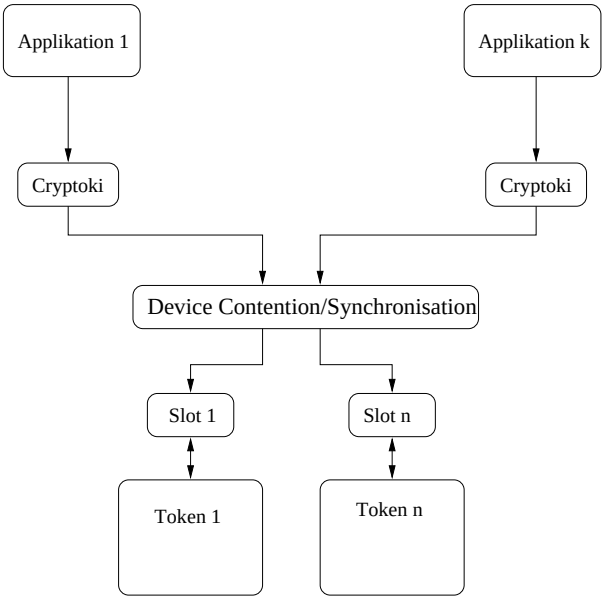

Abbildung 4.2: PKCS #11 aus [15]

Kapitel 5

Übersicht Chipkartensysteme

Hier folgt eine Übersicht einiger Chipkarten, deren Funktionalität und Preis.

Name/ Hersteller	kompatibel zu	Funktionalität	Preis
TCOS-Chipkarten/ Telekom	ISO 7816, SigG zertifiziert E4	RSA 512/1024 Bit, 3-DES, IDEA, Generieren von DES-Schlüsseln, 16K EEPROM	29
GPK8000/ Gemplus	OpenCard, PKCS #11	RSA bis 1024 Bit, Generieren von RSA Schlüsselpaaren	??
GemXpresso 211PK/ Gemplus	JavaCard, Visa Open Platform, PKCS #11	JavaCard 2.1, 24K-EEPROM	??
GemSAFE/ Gemplus	PKCS #11	RSA bis 1024 Bit, 4K EEPROM	??
Sm@rtCafé/ Giesecke & Devrient	ISO 7816, JavaCard	JavaCard 2.1, Generieren von Zufallszahlen 10K EEPROM	25
STARCOS-SPK/ Giesecke & Devrient	ISO 7816, SigG zertifiziert E4	Generieren von RSA Schlüsselpaaren, RSA bis 1024 Bit, DSA bis 1024 Bit, SHA-1, 3-DES, 32K EEPROM	(28)
Cyberflex Access/ Schlumberger	ISO 7816, JavaCard, PKCS #11	JavaCard 2.1, 16K EEPROM	27
Cryptoflex/ Schlumberger	ISO 7816	SHA-1, MD-5, RSA bis 1024 Bit, 3-DES, 8K EEPROM	25
Micardo/ ORGA	ISO 7816, SigG zertifiziert E4	RSA 1024 Bit, DSA 1024 Bit, DES, 16k EEPROM	(32)
Java IButton/ Dallas Semiconductor	JavaCard, OpenCard	JavaCard 2.0, 6K-128K EEPROM	47-127

Tabelle 5.1: Übersicht einiger Chipkarten

Zu Tabelle 5.1:

- Preise in DM bei Abnahme von 400 Einheiten, 1 DM = 1,9566 US $;
- Preise in Klammern: Modell noch nicht verfügbar, Preis des Vorgängermodells;
- Bei der Starcos SPK und der Micardo Chipkarte ist die Zertifizierung nach Signaturgesetz in Arbeit;
- Der Java IButton von Dallas ist nicht - wie die anderen vorgestellen Chipkarten - eine ISO 7810 Plastik-Chipkarte, sondern ein Batteriezellen ähnlicher Metall-Knopf, der als Schlüsselanhänger, als Ring um den Finger oder eingearbeitet in das Armband einer Uhr[1] Verwendung findet. Er verfügt jedoch über eine JavaCard 2.1 Virtual Machine. Der IButton benötigt natürlich ein spezielles Lesegerät, den „Receptor“, und kann mit den anderen in Tabelle 5.2 betrachteten Lesern nicht verwendet werden.

Name/Hersteller	**kompatibel zu**	**Abmessungen**	**Gewicht**	**Preis**
Chipdrive Micro/ Towitoko	CT-API, PC/SC	66x48x11	50	69
GCR 410/ Gemplus	PC/SC, OpenCard	86x85x26	??	135
Reflex 72/ Schlumberger	CT-API, PC/SC	75x70x25	??	88
CardMan Mobile/ Utimaco	CT-API, PC/SC, OpenCard, SigG zertifiziert E2	??	150	200
IButton Receptor Kit/ Dallas	OpenCard	ca. 16mm Ø	??	>29

Tabelle 5.2: Übersicht einiger Leser

[1] s. hierzu http://www.ibutton.com

Zu Tabelle 5.2:

- Preise in DM, bei Abnahme von 200 Einheiten, 1 DM = 1,9566 US $, Gewicht in g, Abmessungen Breite x Tiefe x Höhe in mm;
- Kompatibel zu OpenCard bedeutet in dieser Tabelle, daß der Hersteller eigene Java-native Treiber für die Karte bereitstellt. Grundsätzlich sollte jeder PC/SC kompatible Leser auch mit OpenCard arbeiten, da in OpenCard eine entsprechende Implementierung von PC/SC als CardTerminalService enthalten ist.
- Alle Leser aus Tabelle 5.2 werden über ein serielles Kabel mit dem Rechner verbunden. Es existieren auch Anschlußmöglichkeiten via PCMCIA-Steckplatz, USB, PS/2, parallel, in die Tastatur integriert oder Floppy-Einschübe (siehe auch [13]). Die serielle Form des Anschlusses hat den Vorteil der Plattformunabhängigkeit, aber dafür den Nachteil der Stromversorgung: bei einigen Lesern muß zusätzlicher Strom z.B. über den PS/2 Anschluß zugeführt werden. Andere Lösungen arbeiten mit zwischengeschalteten Akkus (z.B. Towitoko).

Kapitel 6

Bewertung

Welcher Chipkartenleser und welche Chipkarte eignet sich am „Besten“ für den Einsatz in HUELKA ?

Da die Aufgabenstellung verlangt, aus einem Java-Applet heraus auf Leser, Chipkarte und deren Software zuzugreifen, fällt zunächst die Wahl der Middleware auf OpenCard. Nur hier kann ohne großen Aufwand - wie dem Einbinden von Native-Funktionen in Java - direkt das Java OpenCard-API programmiert werden. Da das Java Development Kit auf einer Vielzahl von Plattformen verfügbar ist, können darüber hinaus OpenCard-Applikationen auf verschiedenen Systemen entwickelt werden.

6.1 Chipkarten

Die einzigen kryptofähigen Chipkarten, für die die Hersteller bereits OpenCard-CardServices Implementierungen (s. Tabelle 5.1) bereitstellen, sind die GPK8000, sowie der Java IButton.

Ich schlage jedoch eine kryptofähige JavaCard vor, da diese Kartentypen verschiedene Vorteile haben: sie sind Multiapplikations-Karten, d.h. in ihrem Funktionsumfang nicht auf reine Kryptographie (wie die meisten betrachteten Karten) beschränkt. Mit wenig Aufwand kann ihre Funktion mit Hilfe der Cardlets beliebig erweitert werden - man denke an den Studentenausweis oder andere Einsatzmöglichkeiten. Darüberhinaus ist die Entwicklung von eigenen OpenCard CardServices (falls nötig) für eine JavaCard recht einfach: unter [25] ist bereits ausführlicher Beispiel-Code vorhanden.

Allerdings gilt bei Chipkarten: ist Konformität mit dem Signaturgesetz unverzichtbar, kommt nur die TCOS, bzw. die noch im Zertifizierungsprozeß befindlichen Starcos SPK und Micardo Karten in Frage. Keine andere Chipkarte ist zur Zeit zertifiziert. Eine Anbindung von TCOS an ein Java Applet ist jedoch

sehr aufwendig.

6.2 Lesegeräte

OpenCard ist inzwischen unter den „gängigen“ Lesern der großen Hersteller weit verbreitet, und dementsprechend ist die Auswahl an geeigneten Lesern groß.

Ein Problem bei der Auswahl eines Lesers (s. Tabelle 5.2) bringt jedoch wieder die Konformität mit dem Signaturgesetz: Utimaco hat zur Zeit als einziger Hersteller seine Leser der „CardMan“-Reihe nach ITSEC E2 zertifizieren lassen. Ist die Signaturgesetz-Konformität erforderlich, muß der Leser aus dieser Utimaco-Reihe sein. Falls diese Konformität nicht entscheidend ist, schlage ich das Chipdrive Mini der Firma Towitoko vor: neben seinem unschlagbar geringen Preis sind vor allem die kleinen Abmessungen bei der sinnvollen Platzierung des Lesers von Vorteil. Für seine Einsatzfähigkeit spricht außerdem die Tatsache, daß sowohl die Dresdner- als auch die Commerzbank das Chipdrive Mini an ihre Kunden für Internet-Homebanking ausliefern.

Ob nach der in Abschnitt 2.5 angesprochenen Gesetzesänderung weitere Chipkarten und Leser von den Herstellern zertifiziert werden, ist fraglich: die Nachfrage nach zertifizierten Systemen wird klein bleiben, denn Massenanwendungen, wie z.B. Homebanking funktionieren schon seit geraumer Zeit auch ohne Signaturgesetz, z.B. Homebanking nur über SSL (Deutsche Bank) oder über nicht zertifizierte HBCI[1]-Software mit nicht zertifizierter Chipkarte (Dresdner-/Commerzbank).

Grundsätzlich sollte man daher m.E. überlegen, ob die Chipkartensysteme nach Signaturgesetz überhaupt zertifiziert sein müssen.
In der „Praxis“ werden bereits täglich ohne Probleme Millionen DM per Internet-Homebanking gebucht, und das alles ohne Signaturgesetz. Daher ist es fraglich, ob HUELKA das Signaturgesetz überhaupt braucht.

Außerdem steht für HUELKA kein geeignetes Trustcenter - sondern „nur“ das Rechenzentrum der Universität - zur Erstellung und Verwahrung von Zertifikaten bereit. Fraglich ist darüber hinaus, ob die verwendete Software, also hier OpenCard, das HUELKA Applet oder gar die Java Virtual Machine 1.2 bei gesetzestreuem Einsatz nicht auch zertifiziert werden müssen.

[1] Home Banking Computer Interface

Kapitel 7

Cyberflex Access

Der einzige Hersteller, dem es gelungen ist, innerhalb von drei Monaten seine JavaCard Entwicklungsumgebung kostenlos (Prei ca. DM 980,-) zur Verfügung zu stellen, ist Schlumberger. Die Firma Giesecke & Devrient bestand auf den Kauf ihres DM 1.250,- teueren Toolkits, während GemPlus zwar sowohl die aktuelle Version Ihrer GemXpresso 211 PK, als auch die (nicht JavaCard) GPK8000 versprach, jedoch nicht lieferte.

Das erhaltene „Cyberflex Access SDK“ besteht aus einem etwa 500 Seiten starken englischen Handbuch, zwei Cyberflex Access Chipkarten, einem Leser Reflex 72, sowie einer CD mit der entsprechenden Software in Form von Treibern für den Leser, der Microsoft PC/SC Middleware („Microsoft SmartCard Ressource Manager“) und natürlich dem eigentlichen SDK. Nach der Installation und zweimaligem Neustart - auch unter Windows NT - ist das SDK einsatzbereit. Zunächst fällt jedoch auf, daß der Index des mitgelieferten Handbuchs nicht mit den tatsächlichen Seitenzahlen übereinstimmt. Damit ist das Handbuch als Nachschlagewerk kaum zu gebrauchen. Daß ein großer Teil des Handbuches aus der Aufzählung der einzelnen implementieren JavaCard Klassen besteht, ist unverständlich: zum Nachschlagen sinnvoller ist die Benutzung der JavaDoc Dateien, die mit dem SDK zusammen auf die Festplatte installiert werden. Dafür kommt im Handbuch die Erklärung der verschiedenen „Login-Stufen“ und die Zugriffsrechte auf Dateien der Karte, sowie deren Änderung zu kurz.

Das eigentliche SDK besteht aus einer Art Dateimanager, mit dem man Dateien und Unterverzeichnisse erstellen, Cardlets aus bereits kompilierten *.class* Files erzeugen und APDUs an die Karte schicken kann. Hinzu kommt die Cyberflex Implementierung der JavaCard Funktionen im *.class* Format. Beschänkt man sich beim Programmieren auf die Funktionen des JavaCard 2.1 Standards, die in der Cyberflex Access implementiert wurden (s.u.), so kann man sicherlich auch die originalen (und kostenlosen) *.class* Files von *www.java.sun.com/javacard* benutzen.

Eine komfortable Art der Programmentwicklung, wie z.B. die der GemXpresso RAD (beschrieben in [29]), oder gar eine JavaBean ähnliche Drag & Drop-Form, gibt es bei Schlumberger nicht: eigene Programme müssen in einer anderen Entwicklungsumgebung programmiert und dann mit Schlumbergers „Dateimanager" auf die Karte gespielt werden.

Schlumberger implementiert auf der Cyberflex Access nur einen Teil des JavaCard 2.1 Standards, es fehlen z.B. Funktionen zur Generierung von Schlüsseln oder die MD5 und RIPE Hash-Algorithmen. Den Aufruf der RSA „sign"-Funktion interpretiert Schlumberger als ein SHA-1 Hashen und darauf folgend ein Exponentieren mit dem privaten Exponenten - dies hat zur Konsequenz, daß Sourcecode für die Cyberflex nicht kompatibel zu anderen JavaCards ist. Dies gilt auch für die Cyberflex Funktion *cryptoUpdate*: dies ist bei der Cyberflex die einzige Möglichkeit zum eigentlichen RSA-Exponentieren - *cryptoUpdate* ist jedoch eine reine Schlumberger Funktion und nicht im JavaCard 2.1 Standard enthalten. Der SHA-1 Algorithmus ist außerdem nicht kompatibel mit anderen SHA-1 Implementierungen, z.B. in OpenSSL. Dies bedeutet, daß gleiche SHA-1 Hashwerte nur auf Cyberflex Karten erzeugt werden können und nicht vergleichbar/benutzbar mit anderen Kryptosystemen sind.

Da Schlumberger keinen Simulator für das Testen von Cardlets mitliefert und eigene - nicht JavaCard 2.1 kompatible - Funktionen benutzt, kann auch der in Suns JavaCard 2.1 Paket enthaltene Simulator zum Testen nicht benutzt werden: zum Ausprobieren einer Änderung ist man so gezwungen das Cardlet auf die Karte zu spielen und zu testen, eine sehr zeitaufwendige Prozedur.

Die Cyberflex Access Virtual Machine enthält auch keinen Garbage Collector, wie z.B. einige Versionen des IButtons. Dies bedeutet, daß alle lokalen Variablen von Prozeduren global definiert werden müssen, da sonst schnell sämtlicher Speicher verbraucht ist. Man kann die „elegante" Programmierweise von Java (mit *new*) nicht beibehalten und wird in eine Art Java-Assembler gezwungen - damit geht der große „Java-Vorteil" verloren.

Offenbar scheint die Cyberflex Access grundsätzliche Probleme mit der Speicherverwaltung zu haben. Die Funktion „setKey", mit der z.B. RSA 1024bit Schlüssel initialisiert werden sollen, kann nicht auf einmal auf den ganzen Schlüssel zugreifen, sondern muß sukzessive drei mal jeweils ein drittel des Schlüssels bearbeiten. Dazu steht jedoch kein Hinweis im Handbuch.

Scheinbar „verschluckt" sich die Cyberflex Access auch mit verschachtelten *new* Aufrufen: im OpenCard Beispiel-Quelltext befindet sich ein JavaCard 2.1 Cardlet Source, das im Konstruktor einiger Klassen weitere Klassen (anderen Typs) instanziiert. So auf die Karte geladen, stürzte das Cardlet mit einem unbekannten Fehler ab. Erst als die Instanziierungen aus dem Konstruktor in die aufrufende Funktion verschoben wurden, lief das Cardlet.

Mit dem SDK gibt es zwei Möglichkeiten, aus *.class* Dateien Cyberflex Binär-

Dateien zu erzeugen: einmal aus dem SDK-Menü selbst, und einmal per Kommandozeilenprogramm. Auch hier kommt es zu einigen Ungereimtheiten: das Kommandozeilenprogramm gibt z.B. beim Benutzen von Strings in einem Cardlet richtigerweise die dazu passende Fehlermeldung aus (in etwa: „keine Strings erlaubt"), während das SDK sich über „can't find class]b.class" beschwert.

Der Reflex 72 Leser hat einen bekannten Fehler in der PC/SC Implementierung. Dies hat zur Konsequenz, daß er im OpenCard-TerminalService für PC/SC nicht funktioniert und mit einer Exception abstürzt. Eine Anfrage an Schlumberger nach aktuellen Treibern blieb unbeantwortet.

Das Schlumberger SDK greift per PC/SC auf Kartenleser zu - leider war es nicht möglich ein Towitoko Chipdrive und einen GemPlus GCR 410 zur Mitarbeit mit dem SDK zu „überreden". Obwohl - überprüft mit PC/SC Test-Utilities - beide Leser korrekt am Ressource Manager angemeldet waren, stürzte das Cyberflex SDK beim Starten reproduzierbar ab - dies deutet darauf hin, daß im SDK nicht PC/SC-konforme Befehle benutzt werden.

Ähnliche Ergebnisse sind beim Aufruf der Schlumberger „WebDemo" zu beobachten: sowohl der Internet Explorer (mit aktueller Virtual Machine), als auch der Communicator mit Java 1.2 Plugin stürzen mit dem Fehler *Unknown Operation* ab.

Für die Programmentwicklung in HUELKA ist das Cyberflex Access SDK nicht geeignet.

Literaturverzeichnis

[1] Deutsche Bundestag. *Gesetz zur Regelung der Rahmenbedingungen für Informations- und Kommunikationsdienste (Informations- und Kommunikationsdienste-Gesetz - IuKDG)*, 1997. BT-Drs. 113/7934, Artikel 3 definiert das „Gesetz zur digitalen Signatur (Signaturgesetz - SigG)", s. http://www.bsi.de.

[2] Bundesministerium der Justiz. *ENTWURF eines Gesetzes zur Anpassung der Formvorschrift des Privatrechts an den mordernen Rechtsgeschäftsverkehr*, 1999.

[3] Dr. Stefan Ernst. *Computer und Recht: Die digitale Signatur und ihre juristische Bedeutung*, 1999. Erschienen in: RZ-News 99/9,10, ISSN 1432-7015, S.9ff, Universität Karlsruhe, Rechenzentrum.

[4] Movement for the Use of Smart Cards in a Linux Enviroment. *MUSCLE Smartcard Developers*, 1999. http://www.linuxnet.com/smartcard/index.html.

[5] Regulierungsbehörde für Post und Telekommunikation. *Bestätigte technische Komponenten*, 1999. http://www.regtp.de.

[6] Regulierungsbehörde für Post und Telekommunikation (RegTP). *Maßnahmenkatalog für technische Komponenten nach dem Signaturgesetz*, 1998. s. http://www.bsi.de.

[7] Regulierungsbehörde für Post und Telekommunikation (RegTP). *Maßnahmenkatalog für Zertifizierungsstellen nach dem Signaturgesetz*, 1998. s. http://www.bsi.de.

[8] Bundesamt für Sicherheit in der Informationstechnik (BSI). *Geeignete Kryptoalgorithmen gemäß §17 (2) SigV*, 1999. Erschienen im Bundesanzeier Nr. 213 - Seite 18.638 v. 11. November 1999, s. http://www.bsi.de.

[9] Rinaldo Di Giorgio. *Smart cards: A primer, Develop on the Java platform of the future.* JavaWorld, 1997. http://www.javaworld.com/javaworld/jw-12-1997/jw-12-javadev_p.html.

[10] Transport Layer Security Working Group. *SSL 3.0 Specification*, 1996. http://www.netscape.com/eng/ssl3/.

[11] Uwe Hansmann. *Erwachsen, SmartCards: Betriebssysteme der Zukunft*, 1998. iX - Magazin für professionelle Informationstechnik 8/1998.

[12] IBM. *OpenCard Framework 1.1.1 Programmer's Guide.* Open Card Consortium, 1999. http://www.opencard.org/.

[13] Litronic Inc. *An Introduction to Smartcards*, 1999. http://www.litronic.com /whitepaper/scbibliography.html.

[14] Universität Karlsruhe. *Globalhaushalt und Kosten- Leistungsrechnung*, 1999. http://www.verwaltung.uni-karlsruhe.de/info/ghh/klr_einfuehrung.htm.

[15] RSA Laboratories. *PKCS #11 v2.10: Cryptographic Token Interface Standard*, 1999. http://www.rsa.com.

[16] Deutsche Telekom, GMD, TÜV, TeleTrust. *CT-API 1.1, Application Independent CardTerminal Application Programming Interface for ICC Applications*, 1996.

[17] Frankreich, Deutschland, Niederlande, United Kingdom. *Information Technology Security Evaluation Criteria*, 1991. http://www.alw.nih.gov/Security /FIRST/papers/criteria/itsec.txt.

[18] Microsoft. *Windows CE: SmartCard Background*, 1999. http://www.microsoft.com/windowsce/smartcard/start/datasheet.asp.

[19] Sun Microsystems. *Java(TM) Remote Method Invocation*, 1997. http://www.java.sun.com/products/jdk/rmi/index.html.

[20] The Institute of Electrical and Inc. Electronics Engineer. *IEEE P1361 / D12, Annex D, S. 217ff*, 1999. s. http://www.ieee.org.

[21] Prof. Dr. Alexander Roßnagel. *Die Sicherheitsvermutung des Signaturgesetzes*, 1998. Neue Juristische Wochenschrift.

[22] Bruce Schneier. *Angewandte Kryptographie : Protokolle, Algorithmen und Sourcecode in C*, 1996. ISBN 3-89319-854-7.

[23] Sun. *Java Card (TM)*, 1999. http://java.sun.com/products/javacard.

[24] Erich Gamma und Richard Helm. *Design Patterns - Elements of Reusable Object - Oriented Software*, 1994. ISBN 0-201-63361-2.

[25] Thomas Schaeck und Rinaldo Di Giorgio. *How to write OpenCard card services for Java Card applets*, 1998. http://www.javaworld.com/javaworld/jw-10-1998/jw-10-javadev.html.

[26] Zhiqun Chen und Rinaldo Di Giorgio. *Understanding Java Card 2.0, Learn the inner workings of the Java Card architecture, API, and runtime enviroment.* JavaWorld, 1998. http://www.javaworld.com/javaworld/jw-03-1998/jw-03-javadev_p.html.

[27] GMD und TeleTrust. *CT-BCS, Anwendungsunabhängiger Card Terminal Basic Command Set für Chipkartenanwendungen*, 1995.

[28] Wolfgang Rankl und Wolfgang Effing. *Handbuch der Chipkarten: Aufbau - Funktionsweise - Einsatz von Smart-Cards*, 1996. ISBN 3-446-18893-2.

[29] Thomas Stober Uwe Hansmann. *Programmierte Chips: Fünf Enwicklungsumgebungen für SmartCards*, 2000. iX - Magazin für professionelle Informationstechnik 1/2000.

[30] Visa. *Visa Open Platform*, 1999. http://www.visa.com/nt/suppliers/open/main.html.

[31] The PC/SC Workgroup. *PC/SC Workgroup Specifications*, 1997. www.pcscworkgroup.com.

[32] The PC/SC Workgroup. *Presentation of the Interoperability specification for ICCs and Personal Computer Systems, Revision 2.0*, 1997. White Paper, www.pcscworkgroup.com.