Vermeidung von Datenspuren bei smartcardba−sierten Authentisierungssystemen

Thomas Hildmann

FSP/PV PRZ Technische Universität Berlin, hildmann@prz.tu−berlin.de

Zusammenfassung

Smartcards dienen nicht nur als Schlüsseltechnologie für digitale Signaturen, sie können auch als Identifikationsmerkmal bei einem Authentisierungssystem eingesetzt werden. Dabei wird das Single−Sign−On (SSO) von vielen Experten als die Killerapplikation für Smartcards angesehen. Leider hinterlassen gängige Authentisierungsmechanismen basierend auf Smartcards Datenspuren, die das Anlegen relativ vollständiger Bewegungsprofile ermöglichen. Diese Profile werden um so vollständiger, je weiter ein organisationsübergreifendes SSO umgesetzt ist.
Müssen wir tatsächlich den Mehrgewinn an Sicherheit durch Einsatz der Smartcard−Technologie mit einem Verlust an Privatsphäre bezahlen?
Im Campuskarten−Projekt an der Technischen Universität Berlin wurde ein System basierend auf Java−Cards, einer gegenseitigen Authentisierung von Hintergrundsystem und Karte sowie einer Verteilung der jeweiligen Zuständigkeiten von Systemen (Separation−of−Duty) entwickelt. Dieses System ermöglicht die Authentisierung von Personen und bietet diesen dabei einen möglichst hohen Schutz ihrer Privatsphäre und als Nebeneffekt eine hohe Ausfallsicherheit.
Ausgehend von einem Überblick über die wichtigsten Anforderungen an das Campuskarten−System an der TU Berlin und dem geplanten hochschulübergreifenden System der Hochschulen Berlin−Brandenburg, wird das erarbeitete Datenmodell für die Chipkarte und das Hintergrundsystem erörtert. Danach wird der aktuelle Stand der Arbeit am Authentisierungssystem vorgestellt und versucht, die einzelnen Protokollabläufe zu kommentieren und deren Relevanz bezüglich der Anforderungen deutlich zu machen. Es folgt eine Übersicht über die z.Zt. bekannten Probleme bei der Umsetzung dieser Lösung sowie ein Ausblick über weitere Arbeiten auf dem Gebiet.

1 Einleitung

Oft stehen die Standardanforderungen an IT−Systeme "Sicherheit" und "Benutzungsfreundlichkeit" im Konflikt zueinander. Selbstverständlich ist es einfacher, ein Programm zu starten und sofort auf Daten zugreifen zu können, anstatt zunächst eine Benutzername−/Passwort−Abfrage beantworten zu müssen. Dieses Dilemma versucht man mit Hilfe von Single−Sign−On (SSO) Verfahren zu entschärfen. Statt sich bei jeder Applikation erneut authentisieren zu müssen, wird nur einmal der Authentisierungsprozess durchlaufen. Danach wird die Identität innerhalb des Systems weitergereicht.

Benutzername−/Passwort−Abfragen haben ihre seit langem bekannten Schwächen. Diese reichen von erratbaren Passwörtern über weitergegebene bis hin zu am Bildschirm befestigten Notizen mit allen Passwörtern des Benutzers. Passwortabfragen allein über verteilte Systeme können nicht als sicher angesehen werden, da diese ohne zusätzliche Verschlüsselung der Übertragungskanäle ausgespäht werden können.

Smartcards bieten hier eine benutzungsfreundliche Alternative. Zu den Nachteilen von Smartcards zählen u.a. entstehende Hardwarekosten für Lesegeräte und die Karten selbst, ein in der Regel zeitintensiverer Anmeldevorgang und die nötige Verwaltung von Karten.

Koppelt man die Einführung von Smartcards jedoch mit einem SSO, so relativieren sich die genannten Nachteile. Dieser Effekt kann verstärkt werden, indem nicht anwendungs− oder anwendungsklassenspezifische, sondern multifunktionale Karten eingeführt werden.

Die Kosten können auf die Sicherung einer größeren Anzahl von Diensten umgerechnet werden, die zeitintensive Anmeldung über die Karte ist nur einmal pro Sitzung nötig und die zentrale Verwaltung der Karten hat den nützlichen Nebeneffekt, dass eine Sperrung einer Karte Auswirkungen auf alle Dienste im SSO−Verbund hat.

Hinterlässt eine Anmeldung mittels Benutzername/Passwort nur Datenspuren wie Uhrzeit des An− und Abmeldevorgangs oder auch Ort, von dem aus die Anmeldung stattgefunden hat auf den Applikationsservern, so ist die Anmeldung über ein SSO mit Hilfe einer multifunk−tionalen Karte kritischer zu betrachten! Hier können an zentraler Stelle die Nutzungsinfor−mationen sämtlicher Applikationen gesammelt werden, die über die Karte geschützt sind.

Ein Ausspionieren der Verhaltens− und Leistungsdaten der Karteninhaber (Arbeitnehmer) ist jedoch nicht nur durch den Arbeitgeber möglich, sondern ggf. auch durch Dritte, die z.B. Netzwerkkommunikationen abhören oder an geeigneten Stellen selbst Daten aus der Karte auslesen und zusammenführen.

Selbst wenn eine Smartcard ausschließlich als Container für Zertifikate genutzt wird, lassen sich relativ einfach Bewegungsprofile erstellen. Bei einer zertifikatbasierten Authentisierung muss die Smartcard zunächst ihre Identität preisgeben. Diese Identität kann beliebig kryptisch kodiert sein. So kann jedes Zertifikat als „Subject“−Namen eine eindeutige Nummer besitzen, die zunächst keinen Aufschluss über die dahinter befindliche Person gibt. Es kann jedoch nicht verhindert werden, dass Aufzeichnungen mit den Zeiten und Orten von Zugriffen auf verschiedene Dienste gemacht werden können. In einem letzten Schritt muss der Angreifer die zunächst anonym gesammelten Informationen dann geeignet personalisieren. Dies kann auf verschiedenen Wegen passieren. So reicht es in der Regel, den Netzwerkverkehr von einem bestimmten Netzknoten abzuhören. Findet eine Anmeldung der Chipkarte mit der ID *x* statt und wird kurz darauf das E−Mail−Konto von Person *a* abgefragt, so kann davon ausgegangen werden, dass Person *a* die Karte mit der ID *x* besitzt. Aufschluss über die ID/Personenzuordnung kann ferner über die Nutzung unverschlüsselter Angaben in HTML−Formularen stattfinden. Denkbar wäre auch ein Angriff, bei dem unter einem Vorwand eine HTTPS−Seite mit der Aufforderung Angaben zur Person zu machen vom Angreifer selbst ins Netz gestellt wird. Dieser Server muss nur einen Schlüssel besitzen, welcher von einem bekannten Trustcenter ausgestellt wurde. Der Wechsel von einem HTTPS−Server zu einem anderen wird zwar von vielen Browsern in der Standardeinstellung angemahnt, i.d.R. wird bei einem Wechsel jedoch vom Benutzer der Zertifikataussteller nicht geprüft, solange es sich um einen von einem bekannten Trustcenter ausgestelltes Zertifikat handelt. Mit diesen beiden Beispielen soll gezeigt werden, dass eine Zuordnung zwischen ID und Person keineswegs un−möglich oder mit extrem hohem Aufwand verbunden ist. Hierin steckt eine ernst zu neh−mende Gefahr, der wir in unserem Projekt begegnen wollten.

2 Anforderungen an die Sicherheitsinfrastruktur

Im März 2000 wurde das "Rahmenpflichtenheft − Chipkartenbasierte Dienstleistungssysteme an den Berliner und Brandenburger Hochschulen" vom Arbeitskreis Campuskarte veröffent−licht [NAGE_2000]. In diesem Dokument werden die Rahmenbedingungen für ein hoch−schulübergreifendes Chipkartensystem definiert, die im weiteren Verlauf des Projektes durch Anforderungen des Datenschutzes, der Personalräte sowie weiteren Anforderungen durch die

Verwaltung der Universitäten und durch die bestehenden IT−Strukturen ergänzt wurden [GEBH_2000].

Die wichtigsten Anforderungen an das Authentisierungssystem werden im Folgenden kurz zusammengestellt:

- Über einen Standard−Web−Browser findet die Anmeldeprozedur sowie die Benutzung der Anwendung statt. Als Basistechnologien stehen hier HTML, Java und spezielle Plugins zur Verfügung.
- Der Installationsaufwand von zusätzlicher Software auf den Client−Systemen soll minimal gehalten werden. Idealerweise ist nur der Treiber für den Kartenleser zu installieren.
- Der Benutzer bleibt authentisiert, bis der Browser beendet, ein konfigurierbares Zeitlimit (Inaktivität) überschritten wird oder bis sich der Benutzer explizit abgemeldet hat.
- Bei Überschreiten des Zeitlimits wird eine erneute Authentisierung gefordert.
- Das Autorisierungssystem verfügt über eine geeignete und leicht anpassbare Benutzer−verwaltung.
- Die Benutzung von Smartcards mit integriertem kryptographischen Prozessor und privaten Schlüsseln, die die Karte nicht verlassen können, ist zwingend erforderlich.
- Benutzung über: PC zu Hause, Rechner im Büro, öffentliches Terminal oder PC−Pool jeweils über Standardbrowser und Smartcardreader.
- Authentisierungssoftware wird als Plugin aufgespielt oder vom Server als signiertes App−let geladen.
- Client−Software für alle gängigen Systeme verfügbar, wie MS−Windows, Linux, Mac OS, Solaris, Net/Open/FreeBSD.
- Die auf der Smartcard gespeicherten Daten werden kategorisiert. Es ist festzulegen, welche Personenkreise bzw. Systeme Zugriff auf die jeweiligen Daten erhalten müssen, um ihrer jeweiligen Tätigkeit nachzukommen.
- Es ist sicherzustellen, dass nur autorisierte Personen bzw. Systeme die jeweils für sie be−stimmten Daten von der Smartcard lesen können.
- Die Campuskarte hat folgende Aufgaben:
  - Dienst− bzw. Studierenden−Ausweis: D.h. visuelle Identifizierung (Vergleich des aufgedruckten Passfotos) z.B. beim Zutritt zu Räumlichkeiten oder beim Nachweis über Berechtigung für Ermäßigungen etc.
  - Elektronischer Nachweis über den Gültigkeitszeitraum des Ausweises: Es muss elek−tronisch nachweisbar sein, dass der Ausweis z.Zt. gültig ist.
  - Elektronische Identifizierung von Hochschulangehörigen und Gästen der Hochschulen über Ordnungsmerkmale. Ein Ordnungsmerkmal kann z.B. aus Matrikel− bzw. Personalnummern sowie einer Hochschulkennzahl und einer Versionsnummer für die Karte bestehen.
  - Nutzung der Karte für den Zugang zu Rechnersystemen.
  - Die Karte soll zur Signierung von elektronischen Dokumenten geeignet sein.
  - Dienstliche Dokumente sollen über einen auf der Karte befindlichen Schlüssel codiert werden können.

Weitere Anforderungen und Aufgaben können den erwähnten Quellen entnommen werden. Diese Zusammenstellung dient nur dazu, einen Überblick darüber zu geben, in welchem Kontext die Campuskarte eingesetzt werden soll und auf welchen grundsätzlichen Anforderungen das Campuskartensystem basiert.

Aus den aufgestellten Anforderungen wird ersichtlich, dass die erarbeitete Lösung keineswegs nur für den Einsatz an Hochschulen relevant ist, da die Anforderungen von allgemeiner Natur sind und auf die meisten IT−Infrastrukturen übertragen werden können.

3 Das Datenmodell

Aus den gegebenen Anforderungen wurde ein Datenmodell entwickelt. Ein Teil der Daten befindet sich auf der Chipkarte, wogegen ein anderer Teil im Hintergrundsystem gehalten wird.

3.1 Daten auf der Chipkarte

Einige wenige Daten sollen elektronisch lesbar auf der Campuskarte gehalten werden, damit es auch Systemen ohne direkte Netzwerkanbindung möglich ist, Informationen über den Karteninhaber zu ermitteln. Es wurde jedoch von der Grundlage ausgegangen, dass so wenig Daten wie möglich auf der Karte gespeichert werden sollten [SUHR_2000].

Das Datenmodell teilt die zugreifenden Systeme in sechs Gruppen, die datenhaltende bzw. kartenausgebende Stelle, dazu gehören Studierenden−, Personal− und Gästeverwaltung, hochschulinterne Nutzer, Nutzer im Hochschulverbund, hochschulnahe Nutzer, hochschulferne Nutzer sowie die Karteninhaber. Der Name des Karteninhabers ist auf der Smartcard elektronisch nicht gespeichert. Allein das Ordnungsmerkmal kann an geeigneter Stelle zur Identifizierung genutzt werden. Da es sich beim Ordnungsmerkmal jedoch auch um ein personenbezogenes Datum handelt, ist auch hier der Zugriff beschränkt. Das Java−Applet auf der Karte hat dabei sicherzustellen, dass nur autorisierte Personen oder Systeme auf die jeweiligen Daten zugreifen können.

Die Tabelle 1 zeigt die verwendeten Methoden−Abkürzungen, die ausgeschriebenen Methodennamen und Erläuterungen zum Verständnis der Methode.

Abkürzung	***Methodenname***	***Erläuterung***
c	create	Datum kann einmalig initialisiert werden, d.h. die Erstellung wird auf der Karte angestoßen (z.B. Schlüsselgenerierung).
w	write	Der Wert kann auf die Karte geschrieben werden.
r	read	Der Wert kann von der Karte gelesen werden.
v	verify	Die Karte kann über den gegebenen Schlüssel eine Überprüfung einer Signatur durchführen.
d	decrypt	Über den gegebenen Schlüssel kann eine Entschlüsselung vorgenommen werden.
s	sign	Mit diesem Schlüssel kann eine digitale Signierung vorgenommen werden.
a	auth	Mit Hilfe dieses Schlüssels kann unter Verwendung eines Challenge−Response−Verfahrens eine Authentisierung durchgeführt werden.

Tabelle 1: Zugriffsmethoden auf Datenelemente der Smartcard

Unter "Hochschulinternen Nutzern" verstehen wir im Sinne des Datenmodells Verwaltungs−instanzen innerhalb der Hochschule, die die Karte ausgegeben haben und im Namen dieser Hochschule Dienste anbieten. "Nutzer im Hochschulverbund" sind entsprechend Dienstleister im Namen einer anderen Hochschule, die Mitglied im "Chipkartenverbund" ist. "Hoch−schulnahe Nutzer“ sind Einrichtungen, wie die Mensen etc., wogegen unter die Kategorie "Hochschulferne Nutzer“ alle anderen Systeme fallen.

Datenelement	Application Profile (AP)	Karten ID (CID)	Zert. Karteninhaber – encrypt (CHE)	Zertifikat Kartneninhaber –sign (CHS)	Zertifikat Karteninhaber – auth (CHA)	Zertifikat Trust Center (CTC)	Date of Expire – Begin (DEB)	Date of Expire – End (DEE)	Liste von DES−Keys (DES)	Ordnungsmerkmal (OM)	Statusgruppe (PS)	Geheimschlüssel (PIN)	Fehlversuchhszähler (FVZ)	Geheimer Schlüssel – encrypt (SKE)	Geheimer Schlüssel – sign (SKS)	Geheimer Schlüssel – auth (SKA)
Sichtbare Methoden	cwr	cr	cwrv	cwrv	cwrv	cwrv	cwr	cwr	cw	cr	cwr	cw	c	cwd	cs	cwa
Datenhaltende Stelle	cwr	cr	cwr−	cwr−	cwr−	cwr−	cwr	cwr	cw	cr	cwr	c−	c	cw−	c−	cw−
Hochschulinterne Nutzer und Nutzer im Hochschulverbund	−−r	−r	−−−−	−−−−	−−−−	−−−−	−−r	−−r	−−	−r	−−r	−−	−	−−−	−−	−−−
Hochschulnahe und hochschulferne Nutzer	−−r	−−	−−−−	−−−−	−−−−	−−−−	−−r	−−r	−−	−−	−−−	−−	−	−−−	−−	−−−
Karteninhaber	−−r	−r	−−rv	−−rv	−−rv	−−rv	−−r	−−r	−−	−r	−−r	−w	−	−−d	−s	−−a

Tabelle 2: Datenelemente, Methoden und Zugriffsrechte der Smartcard

Ein Beispiel zum Lesen des Datenmodells:

Das Ordnungsmerkmal (OM) besitzt die Methoden "create" und "read", die nach außen von der Karte zur Verfügung gestellt werden (sichtbare Methoden). Dabei darf die Methode "create" nur von der jeweiligen datenhaltenden Stelle genutzt werden. Das Ordnungsmerkmal kann nach der Initialisierung von allen Zugriffsgruppen, außer von hochschulnahen oder hochschulfernen Nutzern gelesen werden. Um das OM auslesen zu dürfen, muss es sich mindestens um ein System innerhalb des Hochschulverbundes handeln.

3.2 Daten im Hintergrundsystem

Neben den Daten auf der Chipkarte selbst werden noch weitere Daten in Serversystemen ge−halten, die über das Internet erreichbar sind. Bestimmte Anwendungen kommen ohne das Hintergrundsystem aus. Hier möchte ich jedoch, wie bereits erwähnt, nur solche Dienste be−handeln, die über das Internet, insbesondere dem WWW genutzt werden können.

Das Campuskartenmanagementsystem (CKMS) besteht aus dem Kartenstatus Abfrage Sys−tem (KSAS) und der Campuskarten Public−Key−Infrastruktur (CKPKI).

Eintrag	*Datentyp*	*Erläuterung*
Kartennummer	Zahl	Eindeutige Nummer der ausgegebenen Smartcard. Entspricht dem Eintrag CID auf der Karte.
Gültig_von	Datum	Ab wann beginnt die Gültigkeit der Karte. Entspricht dem Eintrag DEB auf der Karte.
Gültig_bis	Datum	Bis wann ist die Karte gültig. Entspricht DEE.
Status	Zahl	Gibt den Status der Karte an (z.B. 0=initialisiert, 1=gültig, 2=gesperrt, ...)

Tabelle 3: Daten im Kartenstatus Abfrage System

Physikalisch könnte das gesamte CKMS auf einem Server gehalten werden. Auch können beide Datenbasen ggf. auch auf einem Datenbanksystem gehalten werden.

Während es sich bei der CKPKI um eine PKI auf Basis von X.509v3 Zertifikaten handelt [GUTM_2000], die über LDAP (Leightweight Directory Access Protocol, [RFC_1777]) abgefragt werden können, handelt es sich bei dem KSAS um ein System, das lediglich Auskunft über den Status einer ausgegebenen Karte gibt. Die Idee dieser Trennung ist die, ein System zu haben, dass zunächst anonym Karten an Hand ihrer Nummer verwaltet und ein weiteres System, das zunächst unabhängig von den Karten Zertifikate verwaltet. Hochschulen, die nur einen Teil der Dienste im Hochschulverbund nutzen wollen und sich z.Zt. gegen den Aufbau einer PKI entscheiden, kommen allein mit dem Einrichten eines KSAS Systems aus.

Hauptvorteil der Lösung ist die Behandlung von Zwischenzuständen, die Mitarbeiterinnen und Mitarbeiter von Hochschulen oft einnehmen. So kommt es häufig vor, dass es zu ungeklärten Arbeitsverhältnissen bei der Verlängerung von Projektverträgen kommt o.ä.. In diesem Fall braucht die Gültigkeit eines Zertifikats nicht verändert zu werden. Es reicht hier aus, den Kartenstatus entsprechend zu setzen. Eine Sperrung des Zertifikats findet so nur dann statt, wenn die Mitarbeiterin bzw. der Mitarbeiter die Hochschule endgültig verlässt oder die Karte als gestohlen bzw. verloren gemeldet wird.

Es fällt auf, dass im KSAS ebenfalls der Gültigkeitszeitraum für die Karte gespeichert wird. Hier entsteht das typische Problem, dass die Daten auf der Karte und im KSAS synchron gehalten werden müssen. Der Vorteil dieses Verfahrens ist, dass nur die Kartennummer bekannt sein muss, um entscheiden zu können, ob es sich um eine zum jetzigen Zeitpunkt gültige Karte handelt. So könnte eine Karte z.B. den Status "gültig" besitzen, der Gültigkeitszeitraum jedoch erst mit dem neuen Semester beginnen.

Beim Entwurf der PKI stellte sich das folgende Problem: Ein Zertifikat, das zur Verschlüsselung von E−Mails oder zur Signierung von Dokumenten dient, muss öffentlich auf einem Verzeichnisserver ablegt werden. Öffentlich zugängige Zertifikate dürfen nicht gleichzeitig Ordnungsmerkmale oder Kartennummern enthalten. Die E−Mail−Adresse gibt in der Regel direkt oder indirekt über den Mailheader Auskunft über den Namen einer Person und im TU−Kontext auch Auskünfte über das Institut bei dem die Person arbeitet oder das Fach, das sie studiert. Diese Information soll z.B. nicht mit der Matrikelnummer verknüpfbar sein, da öffentlich ausgehängte Zensurenlisten i.d.R. die Note und die Matrikelnummer enthalten. Aus diesem Grund wird ein Authentisierungszertifikat gebraucht, das weder E−Mail−Adresse, noch Namen, dafür aber die Kartennummer enthält. Wie später ersichtlich, lässt sich so zum einen sicher die Kartennummer ermitteln, zum anderen bleiben Name und Ordnungsmerkmal aber anonym.

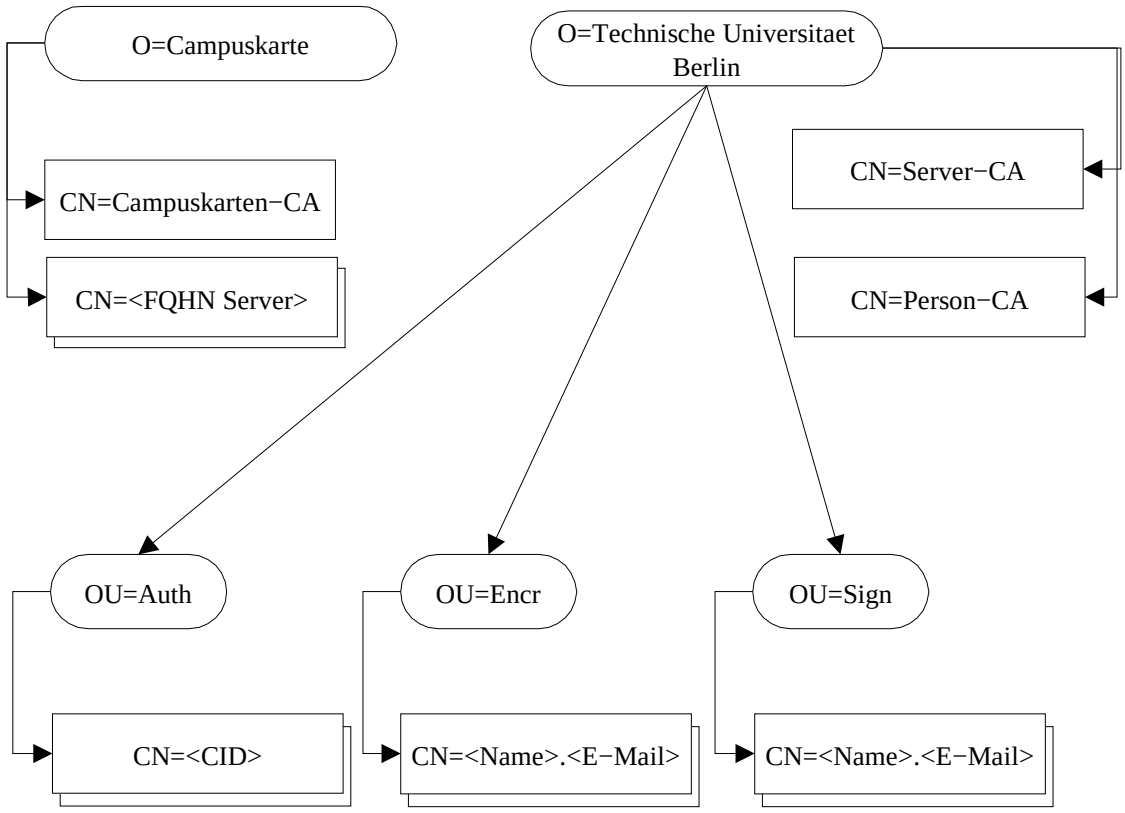

Abbildung 1: Verzeichnisstruktur der Campuskarten PKI

Die drei Zertifikattypen werden in drei Unterbäumen der PKI gehalten (Abb.1). Der Zweig mit den Authentisierungszertifikaten (OU=Auth) ist zugriffsbeschränkt. Er kann nur von Authentisierungssystemen abgefragt werden.

Zu bemerken ist, dass der "Hochschulverbund Berlin−Brandenburg" (O=Campuskarte) und jede Hochschule für sich eine eigene "Server−CA" (CA: Certification Authority, [CAMP_2000]) besitzen. Es wird ferner zwischen "Server−CA" und "Person−CA" unterschieden. Der Grund hierfür ist ein rein technischer:

Während in der Regel für die CA−Schlüssel 2048 Bit RSA−Schlüssel eingesetzt werden, sind die meisten der z.Zt. erhältlichen Smartcards nur zur Überprüfung von 1024 Bit RSA−Schlüsseln in der Lage.

Die Smartcard selbst ist nicht in der Lage, LDAP−Anfragen durchzuführen. Sie kann Zertifikate nur gegen ein zuvor gespeichertes CA−Zertifikat prüfen. Wie später beim Authentisierungsverfahren im Detail zu sehen, ist durch die Trennung "CA", "Server−CA" und "Person−CA" bereits eine Kategorisierung der Zertifikate vorgenommen worden, die die Operationen auf der Karte signifikant reduzieren. Das ist nötig, weil Operationen auf der Karte sowie Übertragungen zur und von der Karte vergleichsweise lange dauern.

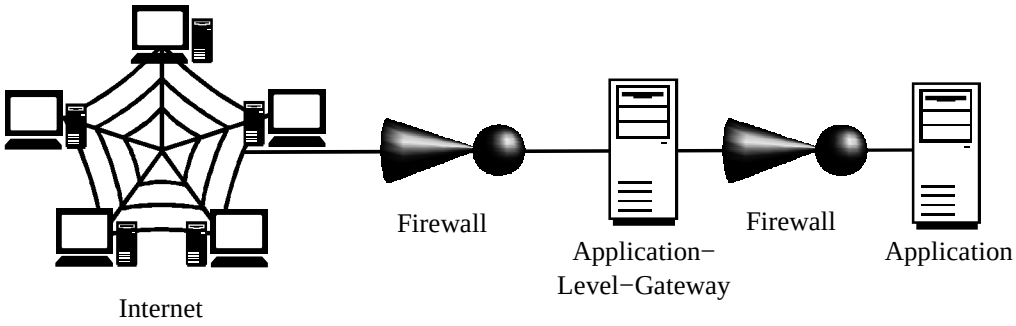

Abbildung 2: Netzwerkarchitektur

4 Die Systemarchitektur

Basierend auf den o.g. Anforderungen und auf dem definierten Datenschema entstand ein Authentisierungsschema, das im folgenden kurz vorstellt und erörtert wird.

Wir haben es in der Regel mit WWW−basierten Anwendungen zu tun, die bereits vor der Einführung der Authentisierung mittels Smartcard entwickelt wurden. Um den Anpassungs−aufwand für die Anwendungen so gering wie möglich zu halten, rüsten wir diese mit Application−Level−Firewall−Gateways aus, die die smartcardbasierte Authentisierung kapseln. Um in gleicher Weise die Sicherheit zu erhöhen, setzen wir anwendungsbezogene Firewalls ein, so dass sich die in Abb. 2 dargestellte Netzwerkarchitektur ergibt.

4.1 Das Client−System

Auf dem Client−System werden ein Browser, ein Smartcardreader und die entsprechenden Treiber sowie eine korrekt konfigurierte Java−Laufzeit−Umgebung vorausgesetzt. Im fol−genden wird davon ausgegangen, dass auf das Client−System ein signiertes Java−Applet geladen werden kann, das dann Zugriff auf den Kartenleser erhält. Dieses Applet wird Identi−ficatore genannt. Es dient als Schnittstelle zwischen Smartcard und Hintergrundsystem.

4.2 Der Application−Level−Gateway

Auf dem Application−Level−Gateway sind fünf Komponenten installiert:

Webproxy: Beim Webproxy handelt es sich um einen einfachen Webserver mit Servlet Un−terstützung, der dazu an Stelle der Applikation vom Browser auf dem Client−System angesteuert wird.

Security−Controller: Der Security−Controller ist ein Servlet, das vom Webproxy für jede Anfrage gestartet wird, die an die Applikation gehen soll. Der Security−Controller stößt die Authentisierung an, stellt für authentisierte Sitzungen Tickets in Form von signier−ten Cookies aus und prüft diese bei jeder eingehenden Anfrage.

Ticketserver: Der Ticketserver dient allein der Generierung und Prüfung von signierten Ti−ckets, die die für die Anwendungen relevanten Daten von der Smartcard enthalten. In der Regel sind das: Ordnungsmerkmal und Personenstatus. Oft kommen hier jedoch noch Daten, wie programminterne Benutzergruppe etc. hinzu.

Decorator: Der Decorator dient der Abbildung der Smartcarddaten (OM und PS) auf die anwendungsspezifischen Daten, wie Benutzername, Benutzergruppe, Sitzungsnummer etc. Der Begriff Decorator ist dem Entwurfsmuster Decorator (Dekorierer, auch Gebundener Umwickler, Wrapper) entliehen, da diese Komponente aus Entwurfssicht genau eine solche Aufgabe übernimmt [GAMM_1996].

Anwendungsproxy: Verschiedene Anwendungen können nicht an die Gegebenheiten der Campuskarteninfrastruktur angepasst werden, weil sie zu schwer zu warten sind oder weil sie z.B. eingekauft wurden und ihr Sourcecode nicht vorliegt o.ä. Um diese Applikationen trotzdem in die Infrastruktur mit einbinden zu können, ist diese Komponente vorgesehen. Sie kann beliebig komplexe, applikationsspezifische Transformationen der Anfragen vornehmen. So kann der Anwendungsproxy beispielsweise gegenüber der Applikation eine einfache Benutzername−/Passwort−Authentisierung durchführen und die Antworten der Anwendung gefiltert an das Client−System weiterreichen. Über den Anwendungsproxy kann später auch eine Anbindung des Systems an ein rollenbasiertes Zugriffskontrollsystem realisiert werden. [GEBH_2000b]

4.3 Weitere Serversysteme

Außer den genannten Komponenten und dem Server, auf dem die Applikation läuft, werden zur Authentisierung ferner ein Authentisierungsserver, das bereits erwähnte Karten Status Abfrage System sowie ein Verzeichnisserver (LDAP) benötigt.

Es ist möglich, die Infrastruktur mit nur einem Authentisierungsserver aufzusetzen. Dies verringert aber nicht nur die Ausfallsicherheit des Systems, sondern stellt am Ende wieder ein Problem bei den anfallenden Logdaten auf diesem Authentisierungsserver dar. Je nach Größe der Organisation sollten mindestens drei Authentisierungsserver eingesetzt werden. Theoretisch können es jedoch beliebig viele Server sein.

KSAS und LDAP Server können auf einem Rechner betrieben werden. Auch hier ist im Sinne der Ausfallsicherheit daran zu denken, für eine möglichst hohe Redundanz zu sorgen, da bei Ausfall eines der beiden Systeme keine Anmeldung mittels Chipkarte mehr erfolgen kann. Eine Verteilung über verschiedene gespiegelte Verzeichnisserver hat ferner den Vorteil, dass über die Anfragen an den Verzeichnisdienst kaum noch Informationen gewonnen werden können.

5 Das Authentisierungsprotokoll

Im Folgenden wird ein Beispielablauf einer Authentisierung gemäß der erarbeiteten Schemata beschrieben; vergl. „Dynamische asymetrische Authentisierung“ [RANK_1999]. Fehlerfälle sowie Spezialfälle u.a. zur Entdeckung von Angriffen gegen die einzelnen Komponenten werden nicht im Detail behandelt. Das Hauptaugenmerk liegt darin, auf welchen Systemen welche Datenspuren hinterlassen bzw. vermieden werden. Ich konzentriere mich in diesem Artikel auf die Frage der Nicht−Verfolgbarkeit bzw. Unverkettbarkeit. Wichtig ist jedoch, das dies nur ein Aspekt des Datenschutzes ist. Weitere Aspekte sind z.B. in [GEBH_2000c] diskutiert.

Die Authentisierung findet in drei Phasen statt, der serverseitigen Authentisierung, der Authentisierung zwischen Smartcard und Authentisierungsserver und der Ausstellung des Tickets.

5.1 Serverseitige Authentisierung

Der Benutzer steuert die gewünschte Anwendung an. Statt direkt mit dem Webserver der Anwendung verbunden zu werden, wird eine Verbindung zum Webproxy hergestellt. Um zum einen sicher zu stellen, dass es sich um den korrekten Webproxy und nicht etwa um ein trojanisches Pferd handelt, und um die spätere Kommunikation zwischen Webserver und Webproxy, also indirekt der Applikation zu sichern, wird zunächst eine SSL (Secure Sockets Layer, [FREI_1996]) gesicherte HTTP Verbindung hergestellt. An dieser Stelle wird über das in SSL definierte Challenge−Response−Verfahren in unserem Fall nur der Server authentisiert. Die Benutzerauthentisierung folgt nun durch das Campuskarten−System.

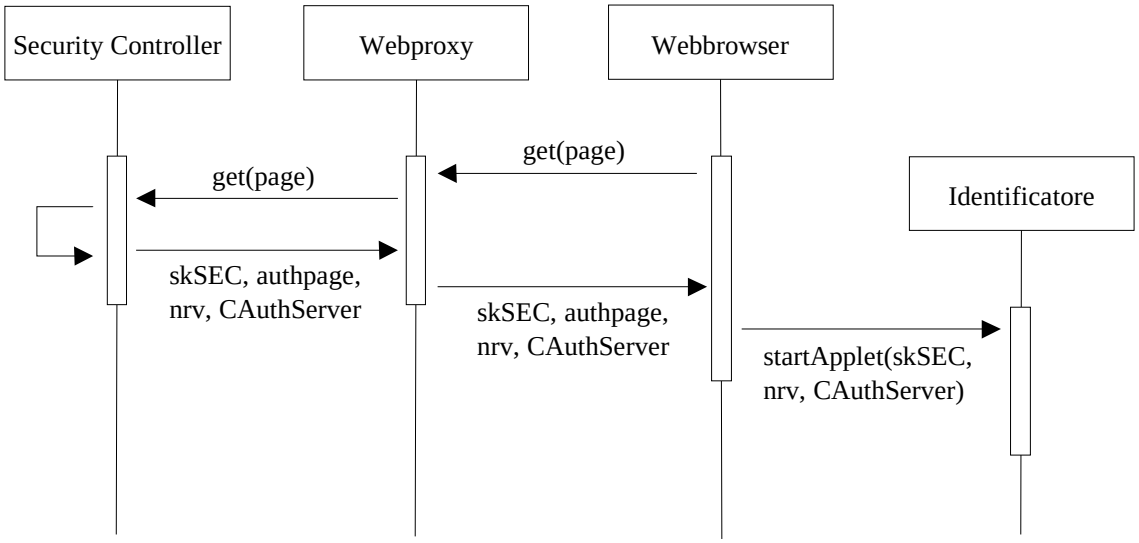

Abbildung 3: Authentisierung Teil 1

Abb. 3 zeigt, wie der Webbrowser eine beliebige Seite *page* anfordert. Die Seitenanforderung wird an das Security Controller Servlet weitergegeben. Der Security Controller stellt fest, dass der Anforderung kein Ticket in Form eines Cookies angefügt ist. Aus diesem Grund startet er den Authentisierungsvorgang.

Der Security Controller generiert einen Sitzungsschlüssel *skSEC* (session key, SEcurity Con−troller). Mit dem Sitzungsschlüssel des Security Controllers werden später Ordnungsmerkmal und z.B. Personenstatus verschlüsselt. Wie wir sehen werden, läuft die weitere Kommunika−tion über den Authentisierungsserver, der jedoch nicht in den Besitz des Ordnungsmerkmals kommen soll.

skSEC wird zusammen mit einer nicht wiederkehrenden Nummer *nrv* und dem Zertifikat ei−nes Authentisierungsservers *CAuthServer* auf einer Authentisierungsseite *authpage* zurück über den Webproxy an den Webbrowser geschickt, der dann das auf der *authpage* angegebene, signierte Applet Identificatore mit den gegebenen Parametern startet.

Die *nrv* ist nötig, um die aktuelle Authentisierungssitzung wiederzuerkennen. Im weiteren Verlauf der Authentisierung kommuniziert das Applet mit dem Authentisierungsserver. Ist diese Kommunikation abgeschlossen, so muss, wie später zu sehen sein wird, der Security Controller eine Möglichkeit haben, das Ticket an den "richtigen" Browser weiterzugeben.

Alternativ könnte *skSEC* zur Identifizierung der Sitzung benutzt werden. Diese Lösung wurde jedoch als "unsauber" empfunden. Die *nrv* identifiziert die Sitzung. *skSEC* dient der Verschlüsselung von Daten, die allein der Security Controller entschlüsseln können soll.

Der Identificatore läuft auf dem Client−System. Da es sich jedoch um ein vom Server geladenes Applet handelt, ist aus Sicht des Benutzers relevant, welche Daten der Identificatore sammeln kann. Aus Sicht der Hochschule muss davon ausgegangen werden, dass mittels gefälschter Java Virtual Machine auf dem Client−System oder durch Ändern des Applets Angriffe vom Client gegen das System gefahren werden können.

5.2 Authentisierung Smartcard/Authentisierungsserver

Es wäre nun möglich, die restliche Authentisierung auch gegenüber dem Security Controller durchzuführen. Hierzu müsste die Karte jedoch ihre CID dem Security Controller mitteilen, um das Zertifikat zu prüfen. Oder die Karte müsste gleich ihr OM preisgeben. Der Security Controller bräuchte irgendeinen Wert, den er im Verzeichnisdienst als Suchmuster benutzen könnte. Die Folge wäre, dass Zertifikate mit Ordnungsmerkmalen abgelegt werden müssten, die man über CIDs suchen kann o.ä.. Auf jeden Fall würden mindestens beim Security Controller CID und OM zusammen kommen.

Da die CID im KSAS benutzt wird, um den Status der Karte zu kontrollieren, aber CID und OM nicht zusammen gebracht werden dürfen[1], wird die CID dem Authentisierungsserver offen gelegt, der jedoch nicht das OM lesen kann, weil dieses mit dem Sitzungsschlüssel für den Security Controller zuvor verschlüsselt wurde.

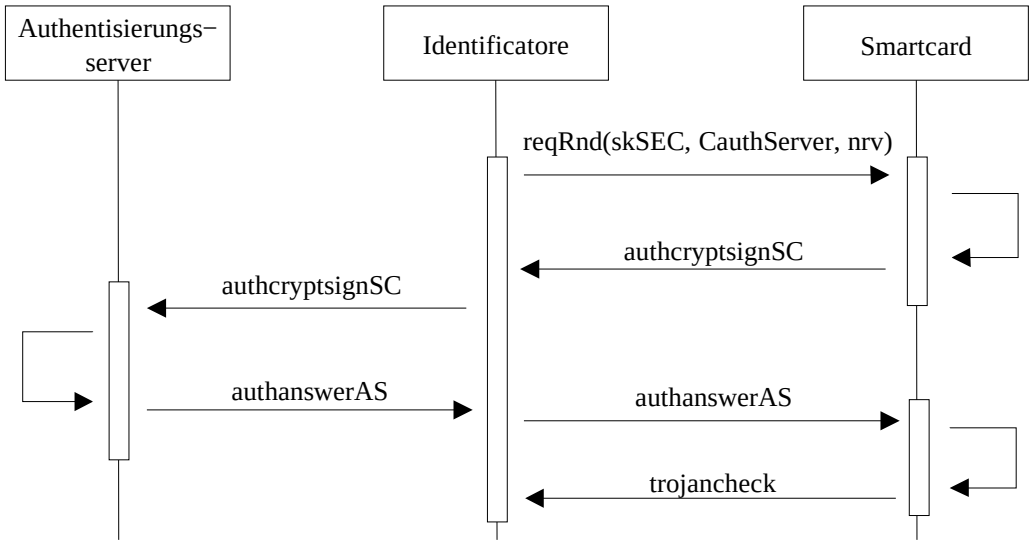

Abbildung 4: Authentisierung Teil 2

Abb. 4 zeigt die zweite Phase der Authentisierung. Phase 1 endete mit dem Start des Identificatore Applets. Dieses Applet ruft nun die Funktion *reqRnd()* mit den Parametern *skSEC*, *CAuthServer* und *nrv* auf.

Die Smartcard prüft zunächst die Korrektheit des *CAuthServer* Zertifikats. Die einzige Möglichkeit, die die Karte besitzt, um eine solche Überprüfung durchzuführen, ist die Verifizie−

[1] Die einzige Ausnahme ist hier die „Datenhaltende Stelle“, weil diese in der Lage sein muss, Karten wieder zu sperren.

rung der Unterschrift der Server−CA des Hochschulverbundes. Es muss sich nicht zwangsweise um ein System der eigenen Hochschule handeln, bei dem sich der Karteninhaber authentisieren will. Hierbei ist zu beachten, dass die Smartcard keine Sperrlisten verwalten kann. Die Kompromittierung eines Authentisierungsservers einer Hochschule muss folglich die Ausstellung eines neuen Server−CA Zertifikats und das Updaten aller Smartcards zur Folge haben. Dies ist jedoch kein Problem unseres Authentisierungsverfahrens, sondern ein generelles Problem bei der Authentisierung eines Hintergrundsystems gegenüber einer Smartcard.

Die von der Server−CA unterzeichneten Zertifikate enthalten u.a. einen Security−Level. Über den Security Level ist bitweise codiert, auf welche Daten das Hintergrundsystem auf der Smartcard Zugriff hat. Die Smartcard extrahiert den Security−Level und prüft, ob das System, für das das *CAuthServer* Zertifikat ausgestellt wurde, dazu berechtigt ist, eine Authentisierung durchzuführen. Ist dies der Fall, wird eine Zufallszahl *rndSC* (RaNDom number SmartCard) generiert und ein zweiter Sitzungsschlüssel *skSC* (SessionKey SmartCard) erzeugt. Danach wird das Ordnungsmerkmal *OM* mit dem Sitzungsschlüssel des Security Controllers *skSEC* verschlüsselt. Das Ergebnis dieser Operation wird in der Variable *omcrypt* abgelegt. Anschließend wird eine Datenstruktur aus *omcrypt*, *rndSC*, *skSC*, *nrv* und *CID* gebildet, die mit dem öffentlichen Schlüssel aus *CAuthServer* verschlüsselt wird. Das Ergebnis der Verschlüsselung wird signiert. D.h. es wird ein Hashwert über den gesamten Datenblock gebildet und dieser mit dem privaten Schlüssel der Smartcard *SKA* verschlüsselt. Das Ergebnis der gesamten Operation *authcryptsignSC* wird über den Identificatore an den Authentisierungsserver gesandt.

Nur der Besitzer des privaten Schlüssels, der zum öffentlichen Schlüssel in *CAuthServer* passt, ist nun in der Lage, *authcryptSC* zu entschlüsseln. Gelingt dies, kann die CID ermittelt werden und somit auch die Signatur über *authcryptsignSC* geprüft werden. Der mögliche Angriff, der durch ein Übertragen eines gefälschten CAuthServer durchgeführt werden könnte, scheitert daran, dass bei der Verifizierung die Unterschrift der Server−CA fehlt.

Der Authentisierungsserver entschlüsselt nun *authcryptSC* mit Hilfe seines eigenen privaten Schlüssels *PrivKeyAS*. Nach der Entschlüsselung wird, zunächst beim KSAS der Status der Karte geprüft. Ist die Karte gültig, nicht gesperrt und innerhalb des Gültigkeitszeitraumes, so wird das Zertifikat *CSc* über die *CID* angefordert. Das Zertifikat wird mit Hilfe des Person−CA Zertifikats *CPersonCA* überprüft. Dann wird der gesamte *authcryptsignSC* Block gegen den öffentlichen Schlüssel aus dem *CSc* geprüft. Sind alle Überprüfungen erfolgreich wird für einen Zeitraum von einigen Minuten das Tupel (*nrv*, *omcrypt*, *rndSC*, *skSC*) gespeichert.

Die Smartcard ist nun in der Lage zu prüfen, ob bei der Entschlüsselung von *authanswerAS* mittels des zuvor generierten Sitzungsschlüssels *skSC* wieder *rndSC* entsteht. Ist dies der Fall, so ist sicher gestellt, dass es sich bei der Gegenstelle wirklich um den Authentisierungsserver handeln muss, da nur dieser im Besitz von *PrivKeyAS* ist, der zur Offenlegung des *skSC* nötig war. Attacken durch Wiedereinspielen vorher aufgezeichneter Sitzungen sind ebenfalls nicht möglich, da sehr wahrscheinlich bei jeder Sitzung eine andere *rndSC* generiert wird. Selbst Wiederholungen der selben *rndSC* können von einem Angreifer nicht erkannt werden, weil diese an den Authentisierungsserver immer im Block mit anderen Zufallszahlen gesendet werden.

Zur Bestätigung sendet die Smartcard einen Auszug aus dem Ordnungsmerkmal, wenn die Prüfung von *authanswerAS* korrekt war.

Zur Zeit ist in der Diskussion z.B. zwei Paare zurückzusenden, die jeweils eine Stelle und die entsprechende Ziffer repräsentieren. Ziel ist, dass der Identificatore eine Meldung auf den Bildschirm bringen kann:

```
Die Authentisierung war erfolgreich!
Zur Gegenprüfung: Die 4. Ziffer ihres Ordnungsmerkmals
lautet 3, die 6. Ziffer lautet 0.
Ist die Gegenprüfung korrekt, klicken sie bitte auf 'Weiter'!
```

Diese Maßnahme soll einen zusätzlichen Schutz gegen trojanische Pferde und gefälschte Webserver bieten. Das Ordnungsmerkmal kann nur von autorisierten Systemen von der Karte gelesen werden. Wird dem Benutzer ein Authentisierungsbildschirm vorgegaukelt, so ist das gefälschte Programm nicht in der Lage, zwei Ziffern des Ordnungsmerkmals zu raten. Für den Benutzer ist dies auf der anderen Seite leicht überprüfbar. Wurde bei der Authentisierung kein Kartenleser mit integriertem PIN−Eingabefeld oder z.B. eine Tastatur mit „secure PIN entry mode“ benutzt, so dass die PIN direkt an die Smartcard gesendet wird und nicht von der Software ausgewertet werden kann, hätte der Trojaner in diesem Fall die PIN des Karteninhabers erlangt. Nur kann diese nicht wieder mit einer Karte in Verbindung gebracht werden, da die Karten allgemein keine Identifikationsmerkmale an nicht autorisierte Hintergrundsysteme ausgeben. Selbst mit falschen Zertifikaten präparierte öffentliche Terminals, die auf gefälschte Webseiten verweisen, können so entlarvt werden.

5.3 Ausstellung des Tickets

Der Karteninhaber wird also vom Identificatore aufgefordert, auf den „Weiter“ Knopf zu drücken.

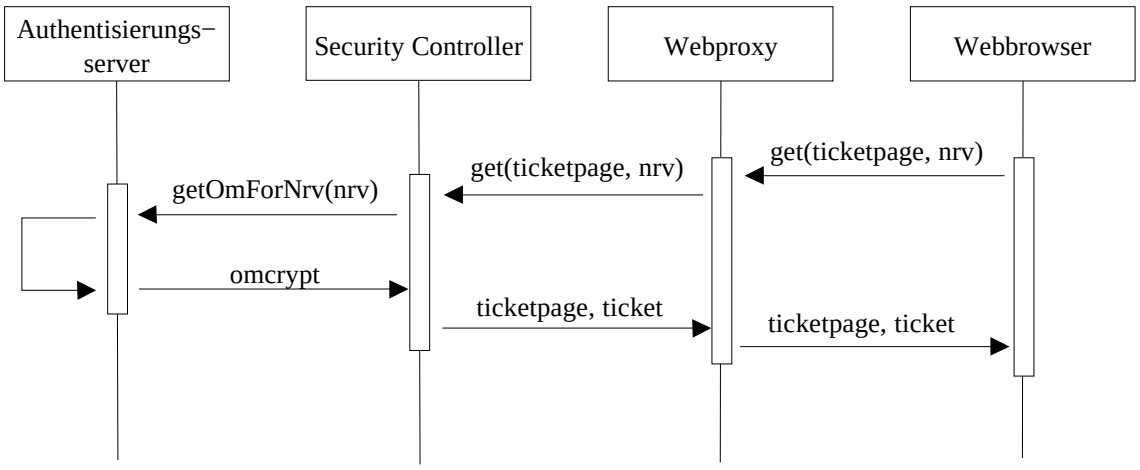

Abbildung 5: Authentisierung Teil 3

Abb. 5 zeigt, wie nach dem Drücken des Weiter−Knopfes die Ticketseite beim Webproxy angefordert wird. Als Parameter wird dem Webproxy dabei die zuvor gesetzte *nrv* mitgegeben. Der Security Controller, der diese Anfrage erhält, fordert nun vom Authentisierungsserver das Ordnungsmerkmal zu der dazugehörigen *nrv*.

Über die *nrv* kann der Authentisierungsserver das *omcrypt* Datum ermitteln und dieses an den Security Controller senden. Da wir Fälle berücksichtigen, in denen Anwendungen auch schreibend auf die Karte zugreifen dürfen (wie z.B. bei der Rückmeldung), übertragen wir auch den *skSC* an die Karte und bewahren diesen Sitzungsschlüssel im Ticket auf. Die Anwendung ist dann in der Lage, ohne neue gegenseitige Authentisierung auf die Karte zuzugreifen, bis die Sitzung beendet ist.

Das Ticket ist dabei vom Ticketserver signiert, so dass sich Tickets nicht von Seiten des Benutzers fälschen lassen. Die SSL−Verbindung zwischen Webproxy und Server stellt sicher, dass keine Daten aus dem Ticket ausgespäht werden können. Bei Ausstellung des Cookies wird der Browser angewiesen, das Cookie nicht zu speichern, sondern nur im Speicher zu halten. Reichen diese Vorsichtsmaßnahmen nicht aus, ist es auch möglich, die Tickets jeweils durch den Ticketserver verschlüsseln zu lassen. In diesem Fall wird dem Benutzer allerdings die Möglichkeit genommen, nachzuprüfen, welche Daten im Cookie über ihn gespeichert sind.

Da der Security Controller im Besitz des Sitzungsschlüssels *skSEC* ist, mit dem das OM verschlüsselt wurde, ist er in der Lage, das Ordnungsmerkmal zu entschlüsseln. In den nächsten Schritten würde dann der Autorisierungsserver angesprochen werden, der zum Ordnungsmerkmal weitere anwendungsspezifische Daten hinzufügen würde. Diese Daten werden dann an den Ticketserver gesandt, der diese serialisiert und signiert. Der Security Controller setzt schließlich die Daten als Cookie und sendet eine Willkommensseite inkl. Cookie an den Browser. Damit ist die Authentisierung abgeschlossen.

6 Transparenz

Aus Benutzersicht ist nicht nur relevant, welche Daten wo anfallen, sondern auch, dass für den Benutzer erkennbar ist, wer welche Daten über ihn gespeichert hat. Leider gibt es keine verlässlichen Möglichkeiten für den Benutzer zu erkennen, welche Daten gerade von seiner Chipkarte gelesen werden. Es wäre zwar denkbar, für verschiedene Daten oder Datengruppen unterschiedliche PIN−Codes zu fordern, so dass der Benutzer eine Kontrolle darüber hätte, welche Daten gerade abgefragt werden. Nur würde das die Karte praktisch unbenutzbar machen. Ob die Daten dann von der Anwendung an Dritte weitergereicht werden, könnte der Benutzer auch in diesem Fall nicht erkennen. Wir entschieden uns aus diesem Grund, nur einen Freigabe−PIN auf die Karte zu schreiben. Wann immer der Benutzer dazu aufgefordert wird diesen einzugeben, ist für ihn zumindest erkennbar, dass nun Daten von der Chipkarte gelesen werden oder kryptographische Funktionen auf ihr angestoßen werden. Um weiterreichende Transparenz zu schaffen, können nur Protokolle, Verfahren und Programmcode offen gelegt werden. Signierung von Applets und Zertifikate von SSL geschützten Webservern können für die Richtigkeit dieser Angaben bürgen.

Selbstverständlich haben die Mitarbeiterinnen und Mitarbeiter sowie die Studierenden die Möglichkeit, über eine eigens hierfür eingerichtete Applikation die Daten, die auf der Campuskarte gespeichert sind, sofern diese ausgelesen werden können, abzurufen und anzuzeigen. Ausnahmen bilden hier die privaten Schlüssel, die die Karte niemals verlassen können, jedoch auch keine weiteren Aussagen über die Person treffen.

7 Fazit

Am Ende der Authentisierung ist der Authentisierungsserver im Besitz der Kartennummer eines Karteninhabers, der IP−Nummer des Clients sowie diverser für die Authentisierung re−levanter, jedoch ansonsten nutzloser Daten wie *rndSC* und *nrv*. Mit dem verschlüsselten Ordnungsmerkmal kann der Authentisierungsserver nichts weiter anfangen. Da sich zur Kar−tennummer jedoch keine personenbezogenen Daten ermitteln lassen, ist eine Protokollierung hier mehr ein statistisches Problem.

KSAS und LDAP−Server kommen lediglich zu Daten darüber, dass die Karte mit einer be−stimmten CID zu einer gegebenen Zeit benutzt wurde. Sie können keine Rückschlüsse auf den genutzten Dienst oder die Quelle der Zugriffe ziehen.

Der Security Controller ist am Ende im Besitz von Ordnungsmerkmal, Zugriffszeit und in Kombination mit den Daten, die der Webproxy gewinnt, auch IP−Nummer des Client−Sys−tems. Unterstellt man jedoch, dass die Zuordnung OM zu Namen bestenfalls in der Anwen−dung gemacht werden kann, so sind die gewonnenen Informationen hier von nicht sehr großer Relevanz. Erst in der Anwendung können im besten Fall Ordnungsmerkmal, Name und Zei−ten zusammen gebracht werden. Dafür geht hier z.B. wieder die IP−Nummer verloren.

Zusammenfassend kann festgestellt werden, dass es mit Hilfe der Teilung des Hintergrundsystems für die smartcardbasierte Authentisierung und durch die Verwendung von Smartcards, die ihre Daten nur autorisierten Gegenstellen aushändigen gelingt, die in dem System anfallenden Datenspuren weit zu verstreuen. Nur durch Zusammenlegen verschiedener Logfiles können Zusammenhänge hergestellt werden. Wird das Aufzeichnen der Daten grundsätzlich an allen Stellen untersagt, an denen eine Protokollierung nicht zwingend erforderlich ist, muss schon von einer kriminellen Vereinigung unterschiedlicher Stellen ausgegangen werden, wenn die Zusammenführung der Daten unterstellt wird.

Das vorgestellte Verfahren ist zeitlich aufwändiger als gängige Verfahren und die Anforde−rungen an die Smartcard sind sehr hoch. Dafür kann dem Karteninhaber und dem Betreiber des Systems in gleicher Weise eine hohe Sicherheit geboten werden.

8 Offene Probleme

Die zur Zeit noch offenen Probleme sind zumeist technischer Art. Dabei stehen Probleme mit dem Zugriff durch die Java Virtual Machine auf die jeweiligen Kartenleser, die verschiedenen Treibersorten sowie die Beschaffung geeigneter Kartenleser im Vordergrund. Auch wären die Nutzer über schnellere Cryptoprozessoren auf der Smartcard glücklich. Es darf nicht verges−sen werden, dass bei der Authentisierung diverse Operationen auf der Karte durchgeführt werden, die alle ihre Zeit brauchen.

Eines der interessantesten Probleme liegt jedoch in der Begegnung der unsicheren Terminals. Manipulationen der Karteninhaber z.B. am Identificatore Applet sind weitgehend wirkungs−los. Was aber ist mit Attacken Dritter gegen die Karteninhaber? Durch Kartenleser mit direkter PIN−Eingabe wird versucht, das Ausspähen der PIN zu umgehen. Was als Angriffspunkt bleibt ist der Browser, der beispielsweise mit falschen Zertifikaten manipuliert sein könnte. Die oben beschriebene Sicherung gegen trojanische Pferde sollte dagegen schützen. Die Praxis wird zeigen, ob die Karteninhaber mit diesem Schutzmechanismen umgehen können oder ob nicht auf anderen Ebenen viel effizientere Attacken gegen das System gestartet werden können (z.B. arbeiten wir z.Zt. daran zu klären, wie sich der Cookie

untrennbar an die Client−Browser−Verbindung binden läßt, so dass es im allgemeinen nutzlos ist, mittels manipuliertem Browser, Cookies zu stehlen).

C. Ellison und B. Schneier stellen in einem Artikel 10 Risiken von PKIs zusammen, denen auf geeignete Weise begegnet werden muss [ELLI_2000].

Hinzu kommt die Frage nach korrekter Software. Hier wurde durch die Anwendung eines objektorienterten Vorgehensmodells, die frühe Suche nach fachlichen Diskussionen, die Verwendung von Unit−Test−Verfahren, Code−Review und Daily−Builds versucht, einen gewissen Qualitätsstandard zu setzen. Zur Zeit ist fast das gesammte Authentisierungssystem in Java implementiert, weil hierdurch die Gefahr von exploitable Buffer−Overflows reduziert werden kann und ferner Code durch alle Instanzen bis hin zur Smartcard wiederverwendet werden kann. Wirklich sicher kann das System allerdings nur durch mehrfache Überarbeitung und Offenlegung, sowie Tests mit großen Benutzergruppen werden.

9 Aussicht

Am 31. Mai wurde der TU−Berlin Öffentlichkeit ein erster Prototyp der Campuskarte und der verwendeten Hintergrundsysteme vorgestellt. Parallel laufen nun Bemühungen, Prüfungs−ordnungen anzupassen, ein Trustcenter aufzubauen, sowie Infrastruktur zur Personalisierung der Karten zu schaffen. Andere Hochschulen haben bereits ihr Interesse an der erarbeiteten Lösung bekundet.

Meiner Ansicht nach dürfte das beschriebene Authentisierungsschema jedoch nicht nur für Hochschulen interessant sein. Ich sehe die Anwendbarkeit überall dort, wo eine multifunk−tionale Smartcard eingeführt werden soll, der Betreiber jedoch darauf bedacht ist, den Kar−teninhabern ebenso weit reichende Schutzmechanismen zu bieten, wie sich selbst.

Chipkartensysteme sollten langsam weggeführt werden von ihrem Image als perfektionierten Nachfolger der Stechuhr oder als letzten Schritt zum Big Brother Staat. Mit dem vorliegen−den Artikel habe ich versucht darzustellen, dass dies mit heutigen Mitteln möglich ist.

Literatur

BANN_2001 J. Banning: *LDAP unter Linux − Netzwerkinformationen in Verzeichnisdiensten verwalten*, Addison−Wesley München 2001

CAMP_2000 I. Camphausen et al.: *Aufbau und Betrieb einer Zertifizierungsinstanz*, DFN−PCA Hamburg, März 2000

ELLI_2000 C. Ellison and B. Schneier: *Ten Risks of PKI: What you're Not Being Told about Public Key Infrastructure*. Computer Security Journal, v 16, n 1, 2000, pp. 1−7. http://www.counterpane.com/pki−risks.html

FREI_1996 A. O. Freier et al.: *The SSL Protocol Version 3.0*, Internet−Draft, Netscape Communications, November 1996 http://home.netscape.com/eng/ssl3/draft302.txt

GAMM_1996 E. Gamma et al.: *Entfurfsmuster: Elemente wiederverwendbarer objekt−orientierter Software*, Addison−Wesley Bonn 1996

GEBH_2000 T. Gebhardt et al.: *Anforderungsmodell*. FSP−PV/PRZ Technische Universität Berlin August 2000.

GEBH_2000b T. Gebhardt und T. Hildmann: *Enabling Technologies for Role Based Online Decision Engines*. Fifth ACM Workshop on Role−Based Access Control, Berlin 2000 S. 77−82.

GEBH_2000c T. Gebhardt et al.: Sicherheitsmodell, FSP−PV/PRZ Technische Universität Berlin Oktober 2000

GUTM_2000 P. Gutmann: *X.509 Style Guide*. October 2000. http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt

LANG_2000 S. Lange: *Angst vor dem gläsernen Studenten.* die tageszeitung Berlin lokal, 17.4.2000 http://www.taz.de/tpl/2000/04/17/a0231.nf/stext.Name,ask08003aaa.idx,1

LANG_2000b S. Lange: *Die Uni lädt zu Chips.* die tageszeitung Berlin lokal, 17.4.2000 http://www.taz.de/tpl/2000/04/17/a0262.nf/stext.Name,ask08003aaa.idx,0

NAGE_2000 K. Nagel (Editor): *Rahmenpflichtenheft − Chipkarten−basierte Dienstleistungssysteme an den Berliner und Brandenburger Hochschulen*. Berlin/Brandenburg März 2000. http://wwwpc.prz.tu−berlin.de/tu−chipkarte/daskonzept/Rahmenpflichtenheft.htm

RANK_1999 W. Rankl und E. Effing: *Handbuch der Chipkarten.* Carl Hanser Verlag München Wien 1999.

RFC_1777 W. Yeonf, T. Howes, S. Kille: *RFC 1777: Lightweight Directory Access Protocol*, 1995

SUHR_2000 L. Suhrbier: *Smartcard−Datenmodell*. FSP−PV/PRZ Technische Universität Berlin November 2000.