Vermeidung von Datenspuren bei smartcardbasierten Authentisierungssystemen

Thomas Hildmann
<hildmann@prz.tu-berlin.de>

Inhalt

1. Der Kontext: Campuskarte, TU-Berlin
2. Scheinbar unlösbar: Identifizierung ohne Verkettbarkeit
3. Ablauf und Komponenten der Authentisierung
4. Technische Realisierung
5. Zusammenfassung

Der Kontext: Campuskarte, TU-Berlin

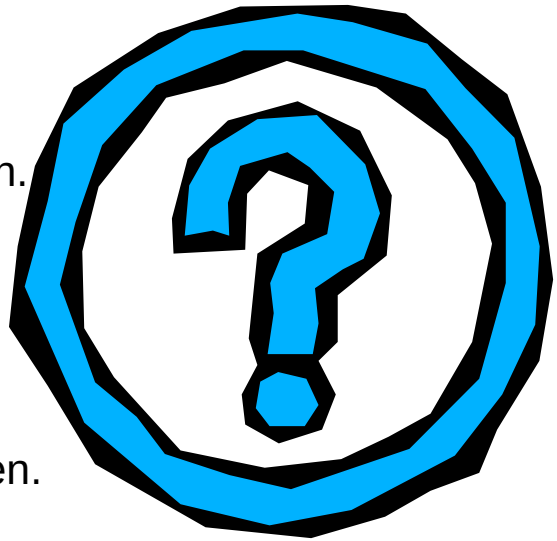
- Standard-Web-Browser
- Min. Installationsaufwand
- SSO mit konf. Zeitlimit
- Smartcards mit integriertem Kryptoprozessor
- Client-Software für MS-Windows, Linux, MacOS, Solaris, *BSD
- So wenig Daten, wie möglich auf der Karte
- Zugriffsbeschränkung über gegenseitige Public-Key-Authentisierung
- Dienst- bzw. Studierendenausweis
- Elektr. Nachweis über den Gültigkeitszeitraum des Ausweises
- Elektr. Identifizierung von
  - Hochschulangehörigen
  - Gästen
- Zugangskontrolle zu Rechnern
- Signierung elektr. Dokumente
- Verschlüsselung dienstlicher Dokumente

Was steht auf der Karte

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 Hoch-schulverbund	--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

Scheinbar unlösbar: Identifizierung ohne Verkettbarkeit

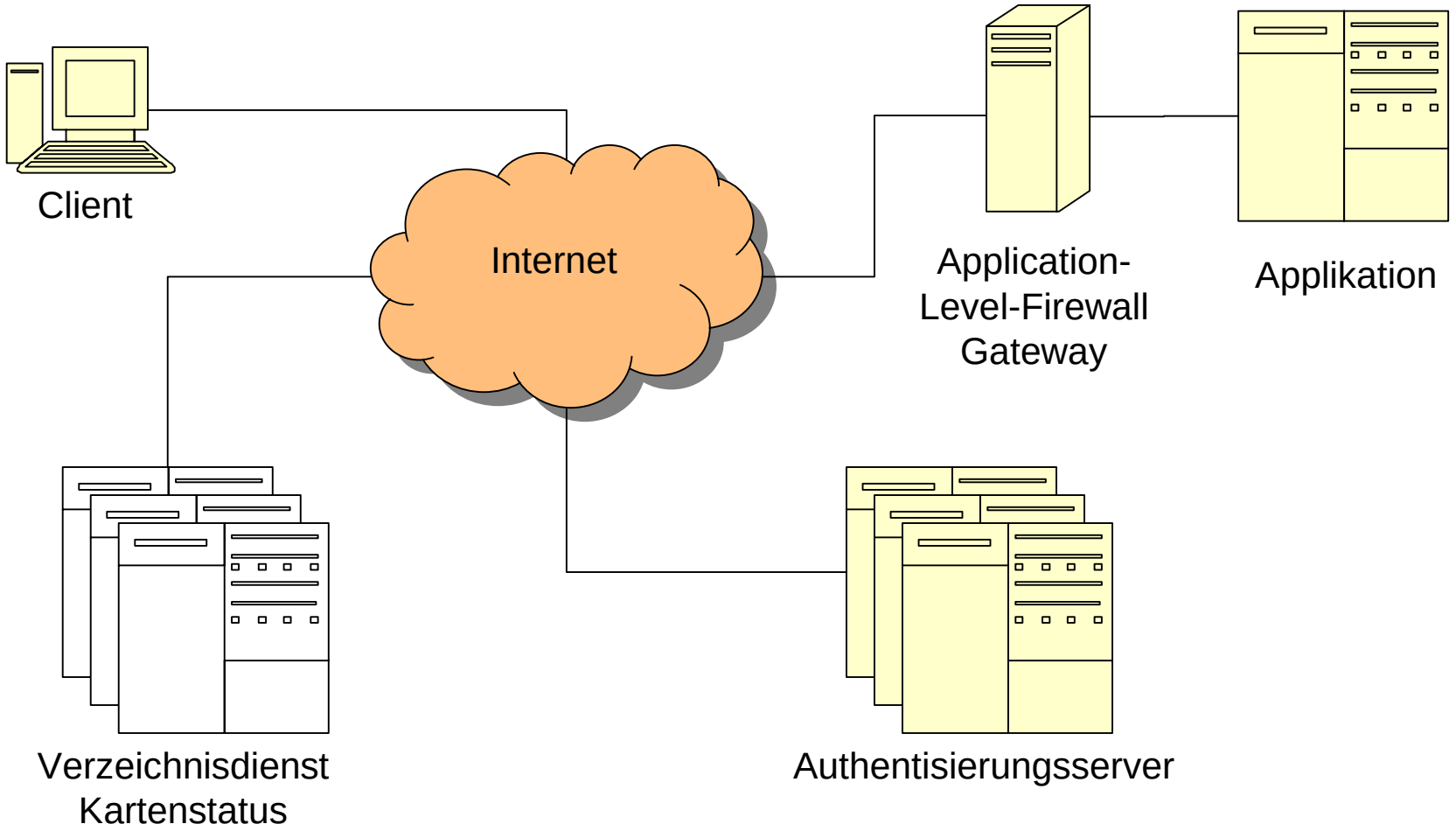
- Benutzer soll eindeutig und unabstreitbar identifiziert werden.
- Es soll nicht ohne weiteres möglich sein, Bewegungsprofile über den Benutzer anzulegen.
- Die Benutzung von Applikationen soll weitestgehend anonymisierbar sein.
- Unauthorisierte Zugriffe sollen verhindert werden. Insbesondere soll zentral gesperrt werden können.
- Schließt sich das nicht alles aus?



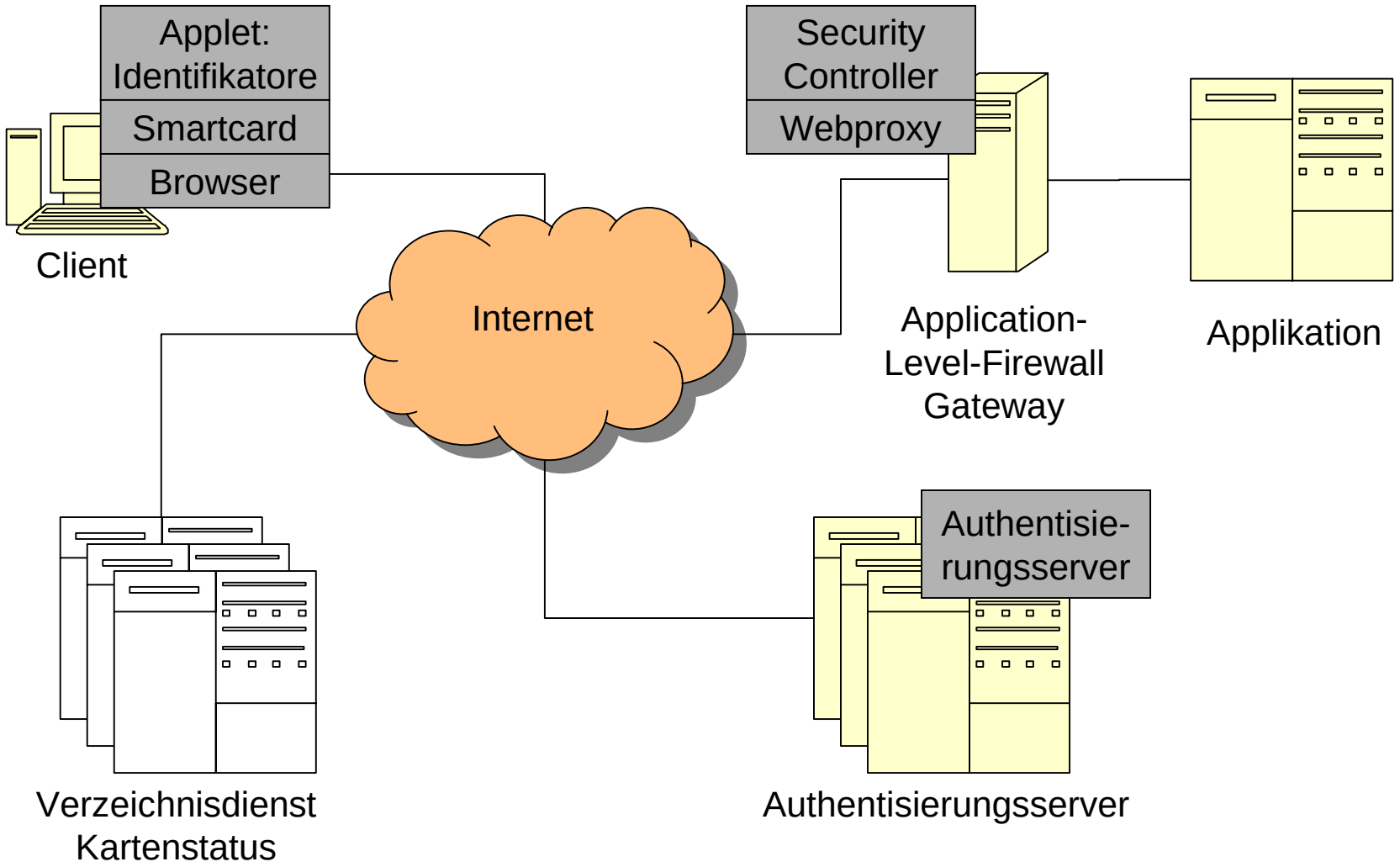
Ablauf und Komponenten der Authentisierung

- Der Lösungsansatz:
  - Die Kartennummer (CID) weist die Identität einer Karte nach.
  - Das Ordnungsmerkmal (OM) weist die Identität einer Person nach.
  - Bei der Personalisierung einer Karte werden CID und OM getrennt.
  - Die Gültigkeit der Karte wird vor dem Authentisierungsserver nachgewiesen. Hier wird auch die Sperrung kontrolliert.
  - Die Identität der Person wird vor dem Security Controller nachgewiesen.
  - Betrieb von Authentisierungsserver und Security Controller erfolgt durch unabhängige Instanzen.

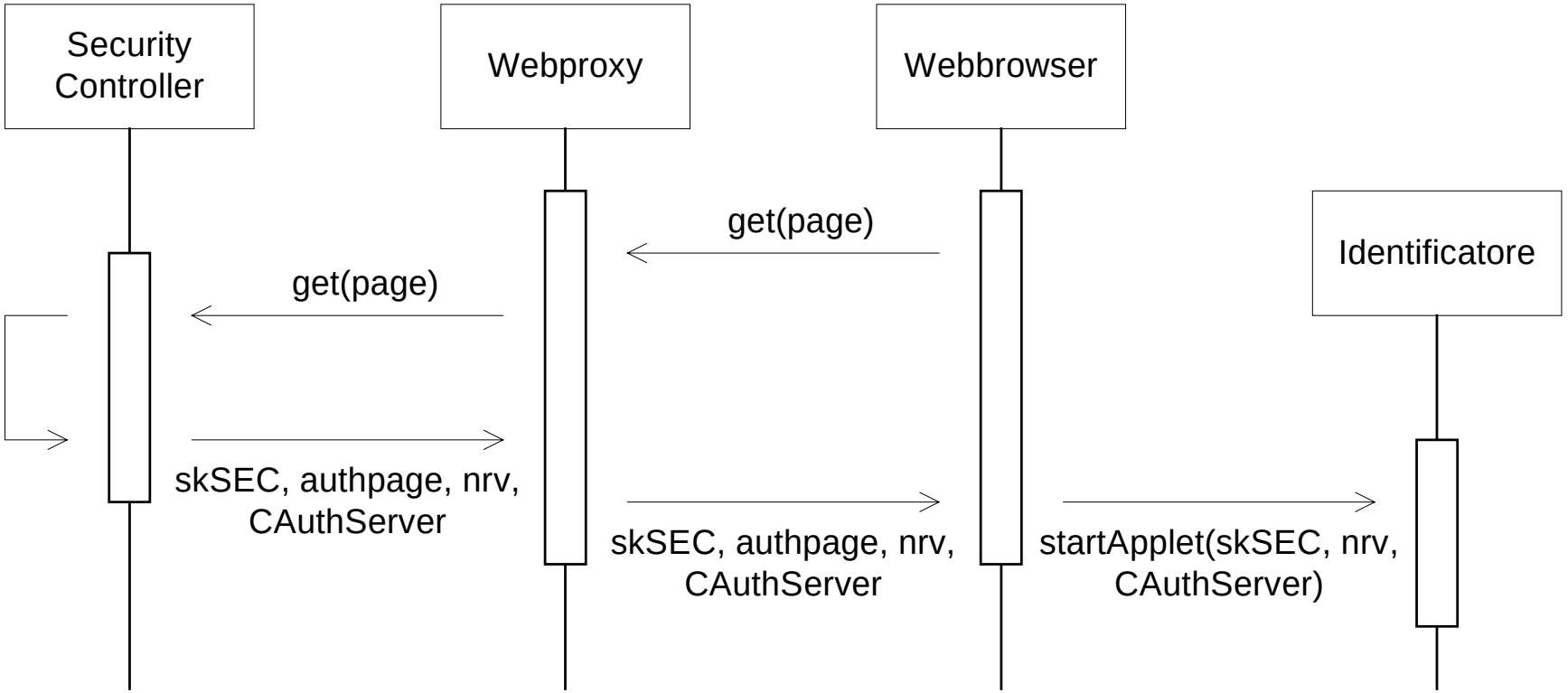
Architektur



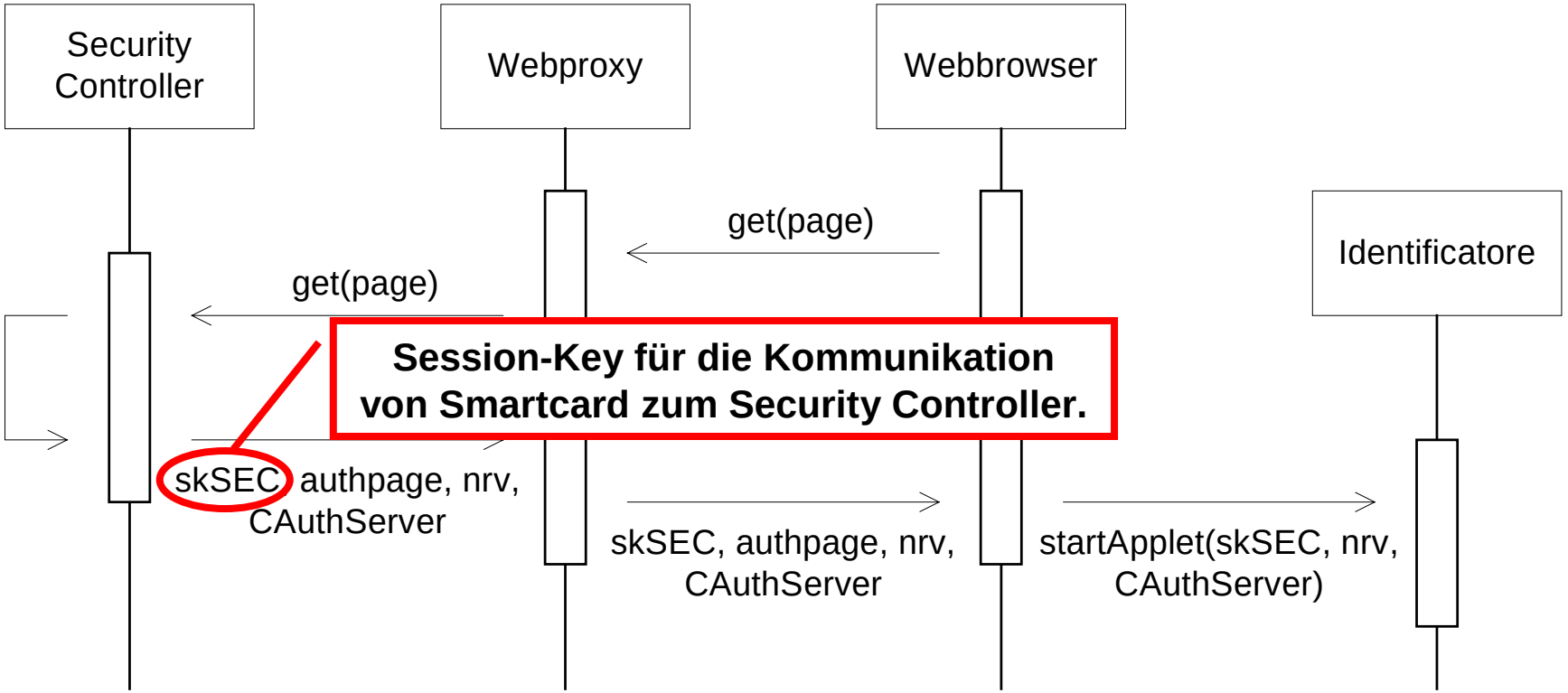
Architektur



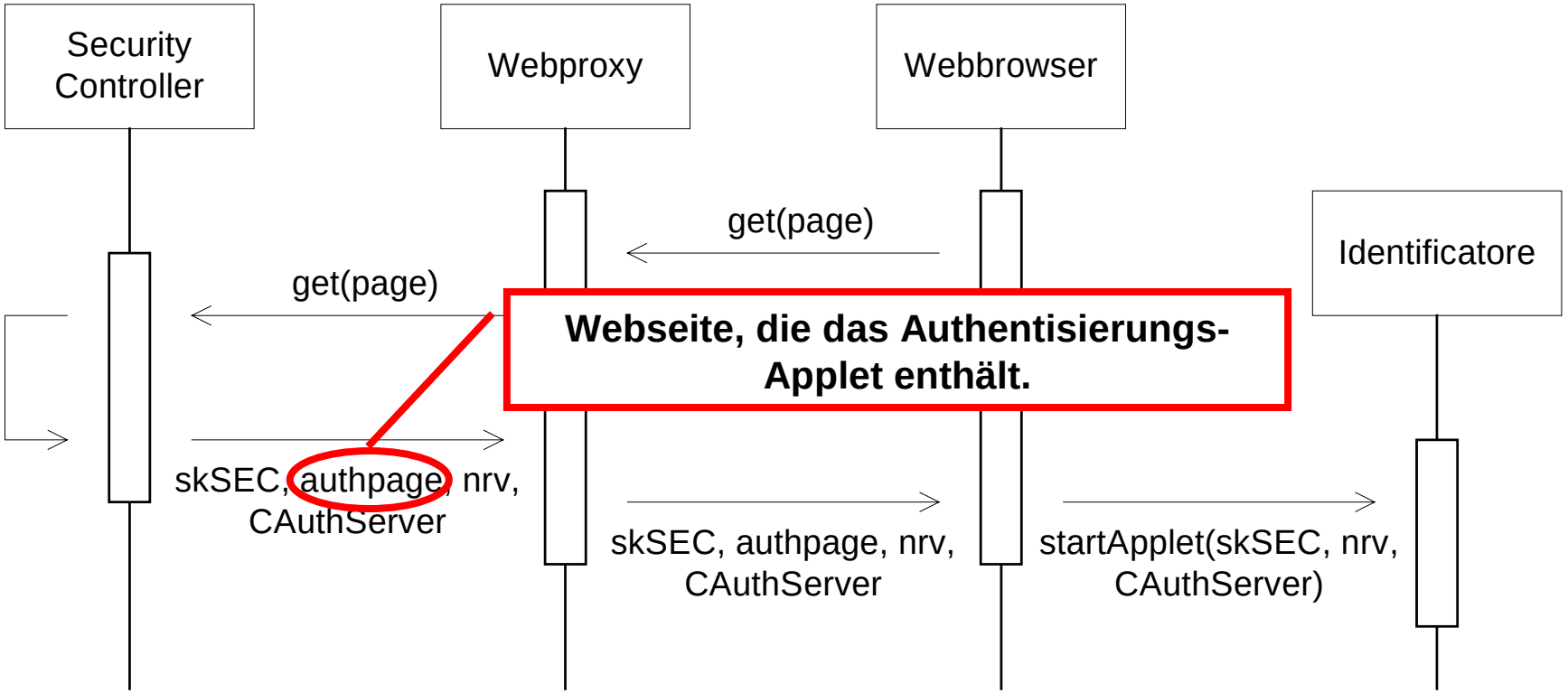
Authentisierung Teil 1



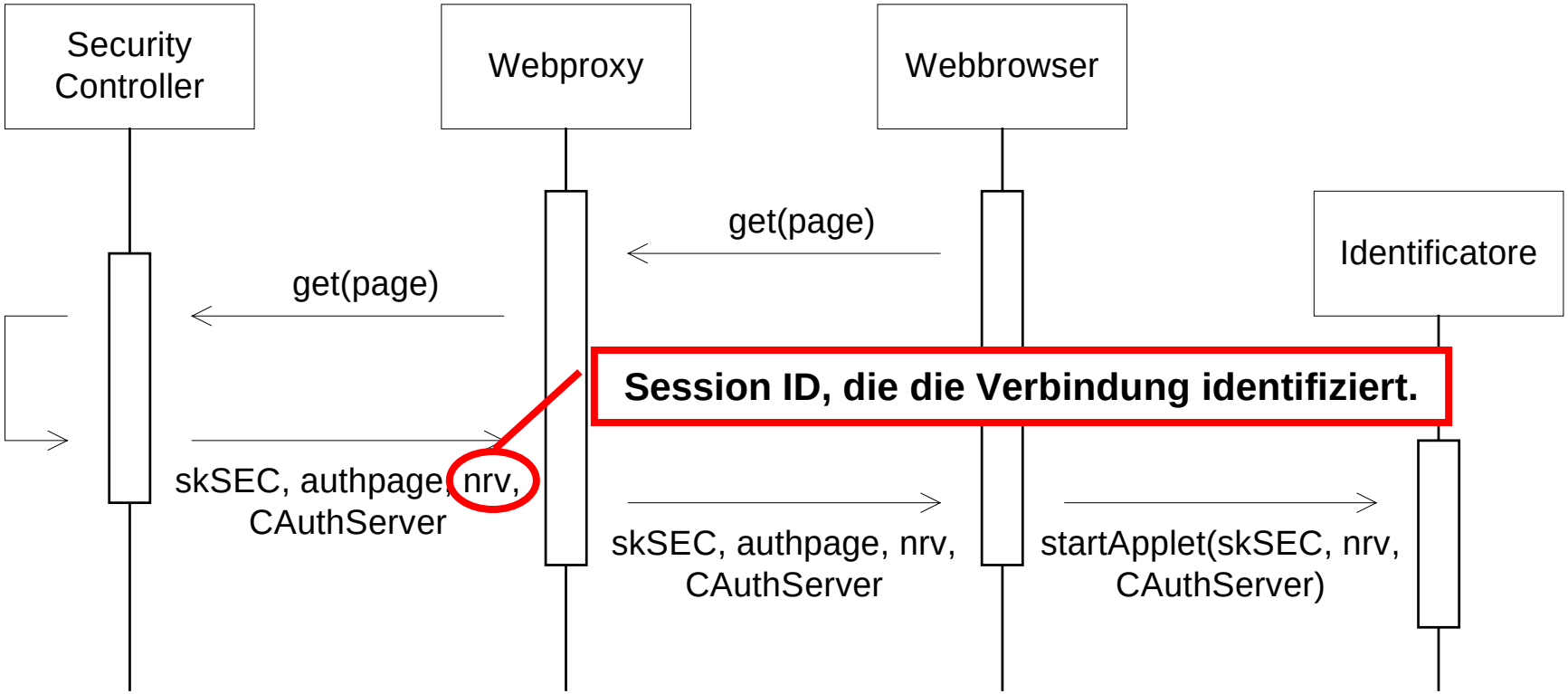
Authentisierung Teil 1



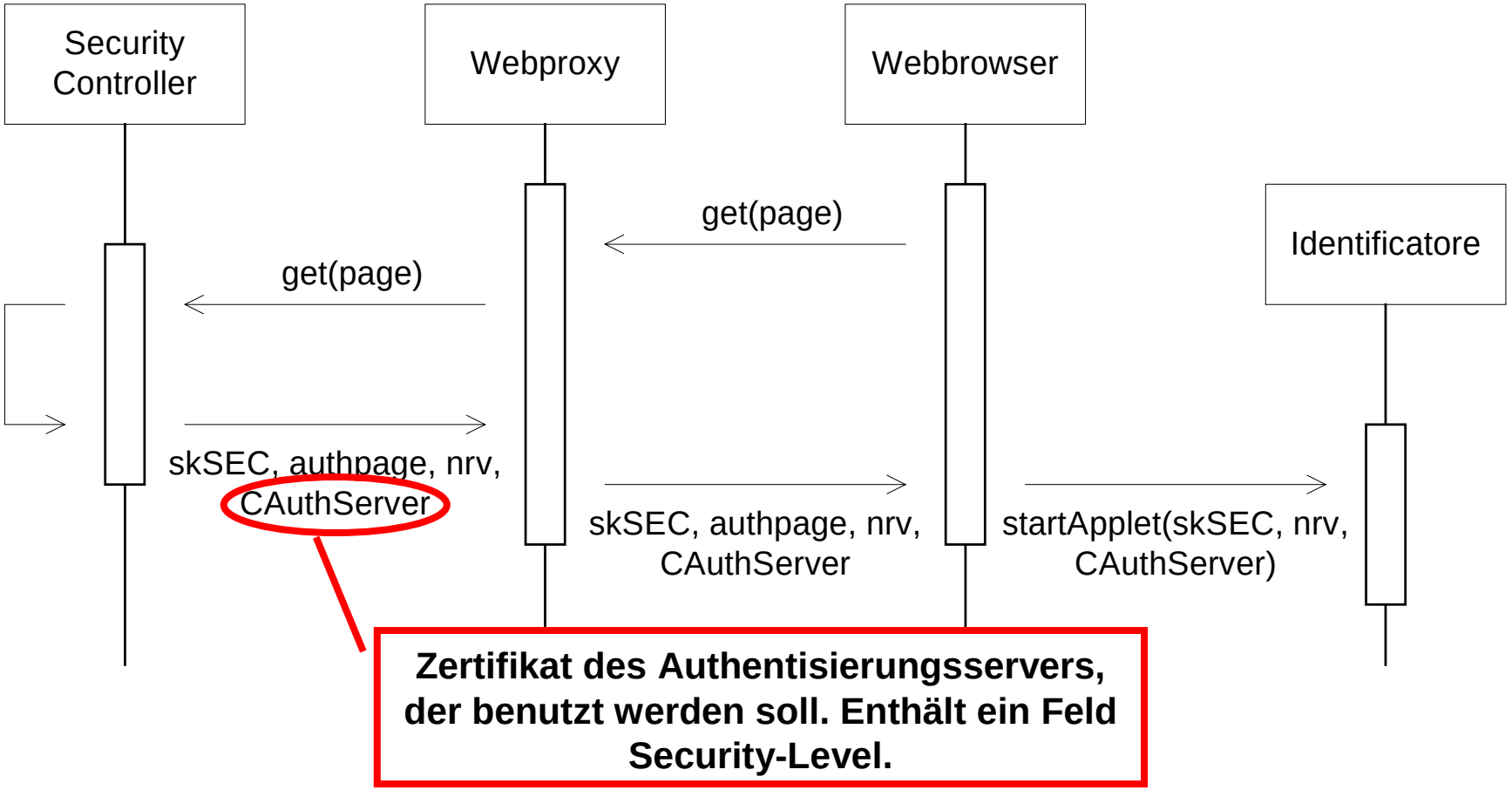
Authentisierung Teil 1



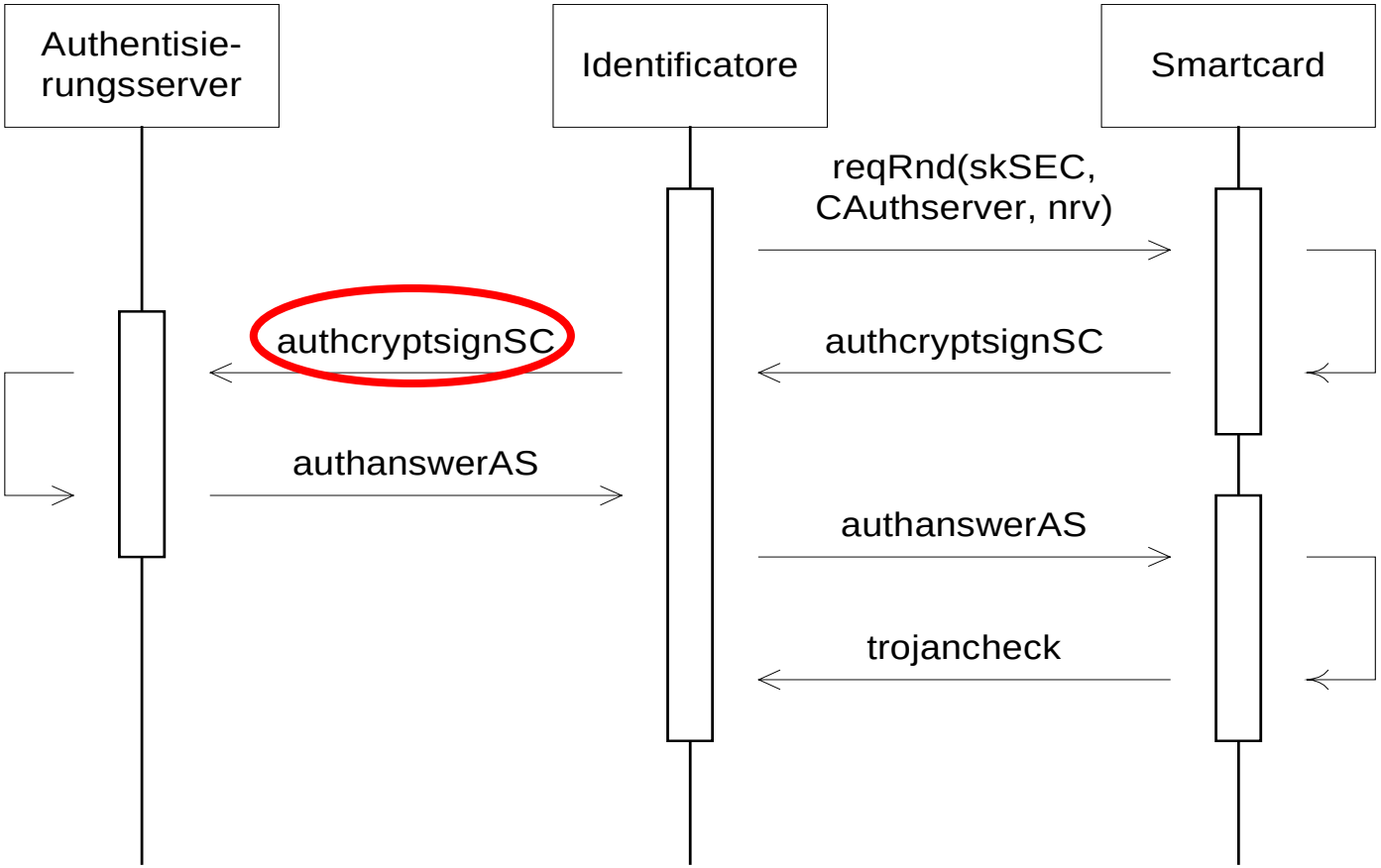
Authentisierung Teil 1



Authentisierung Teil 1



Authentisierung Teil 2

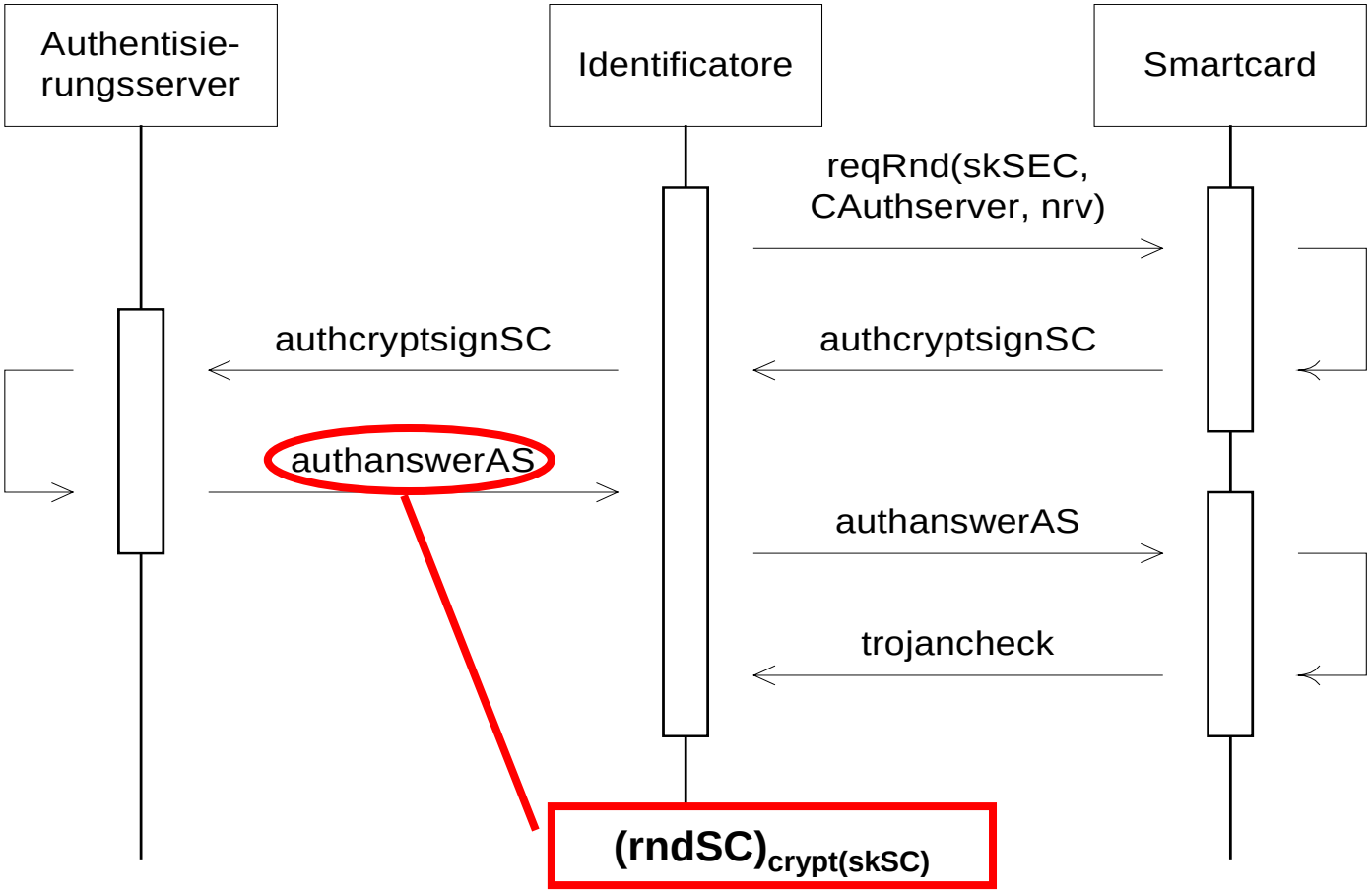


Inhalt von authcryptsignSC

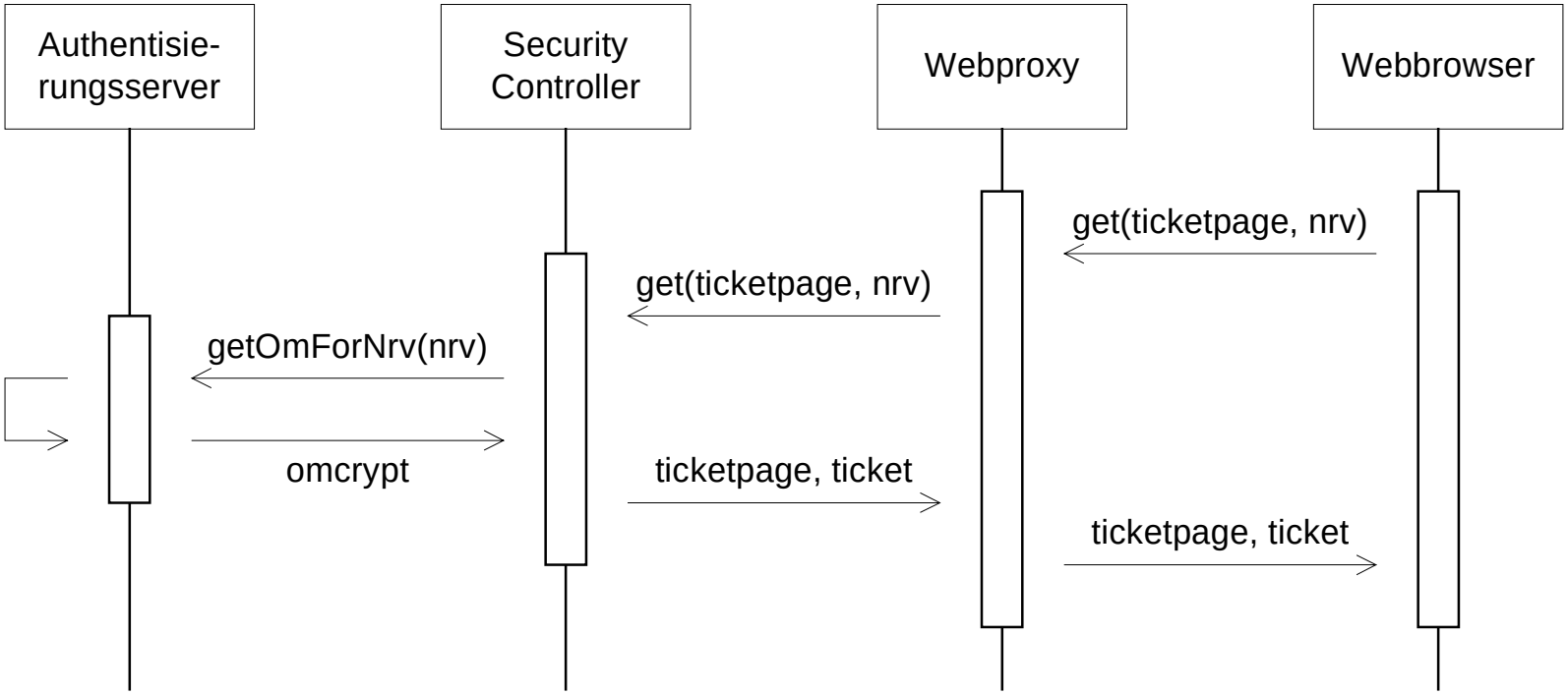
- *rndSC*: Zufallszahl der Smartcard
- *skSC*: Von der SC erzeugter Session-Key
- *omcrypt*: Das Ordnungsmerkmal verschlüsselt mit dem Session-Key des Security Controllers (nicht lesbar für den Authserver!)
- *CID*: Die Card Identification Number

$$authcryptsignSC = ((omcrypt, rndSC, skSC, nrv, CID)_{crypt(AuthServer)})_{sign(SC)}$$

Authentisierung Teil 2



Authentisierung Teil 3



Technische Realisierung

- **Smartcard**: Java-Card mit Crypto-Co-Prozessor
- **Authserver**: Linux/BSD-Server, Java-Application
- **Identificatore**: Java-Applet
- **Webproxy**: Linux/BSD-Server mit Apache
- **Security-Controller**: Servlet
- **LDAP**: Linux-Cluster mit OpenLDAP 1.x
- **KSAS**: Python-Script auf LDAP-Cluster

Ausblick

- In den letzten Jahren: RBAC
- Gerade Arbeit an Smartcards.
- Nächster Schritt ist die Kombination von beidem.
  ➔ s. Session 1 von gestern zum Thema: Identitätsmanagement

Zusammenfassung

- Verschleierung von Bewegungsprofilen trotz sicherer Identifizierung mittels Smartcards ist möglich.

- Java-Cards lohnen sich hier, auf Grund der Möglichkeit, komplexere Verfahren zu implementieren und anzupassen. Dabei wird eine Herstellerunabhängigkeit gewahrt.

- Durch die Verwendung von Java als Implementierungssprache auf den Smartcards wird die Möglichkeit einer unabhängigen Begutachtung des Codes stark vereinfacht.

Dank an Mitwirkende

- Leitung
  - Klaus Nagel
  - Klaus Rebensburg

- **Entwicklung und Konzeption**
  - **Shpresa Dafoli**
  - **Thomas Gebhardt**
  - **Klaus Hamann**
  - **Carsten Kudwien**
  - **Lutz Suhrbier**

- Realisierung / Programmierung
  - Christian Achter
  - Andrea Rebecchi
  - Christoph Wöller
  - Sebastian Zickau

- Verwaltung und Öffentlichkeitsarbeit
  - Gisela Krieg
  - Klaus Oberzig
  - Daniel Pruss

- u.a.

Zum Artikel selbst

- Der Artikel ist sehr umfassend und technisch relativ Detailliert.
  ➔ Entschuldigung dafür!

- Ziel war es, die Idee auf diesem Detaillierungsgrad zu sichern, weil zum Zeitpunkt der Erstellung des Artikels, die rechtliche Grundlage zur Verwertung der Ideen unklar war.

- Weiterhin war es mein persönliches Ziel, einer Patentierung des Verfahrens vorzugreifen und das Verfahren nicht mehr patentfähig zu machen.

- Fehler: Im Artikel steht einige Male „Server-CA“, wo eigentlich die Campuskarten-CA gemeint ist. Die Campuskarten-CA erstellt die Zertifikate der Authentisierungsserver.