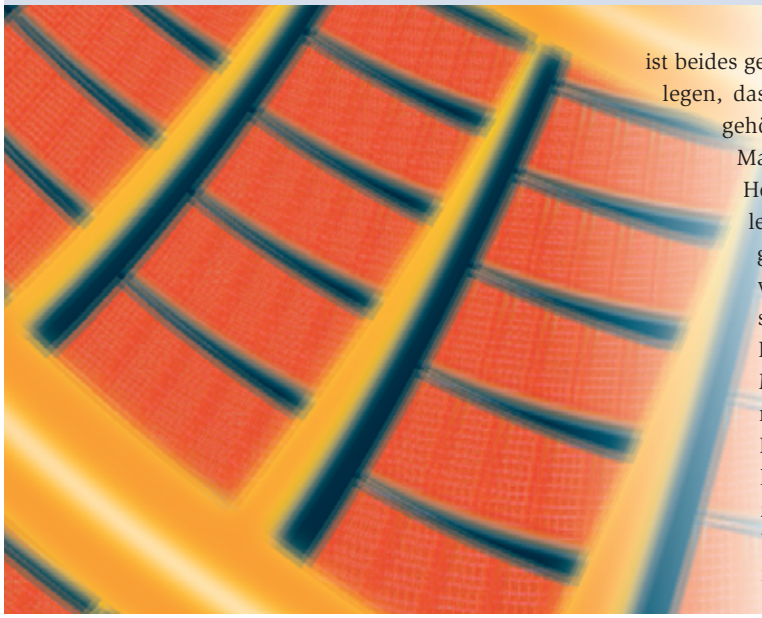
Grundlagenwissen: Authentifizierungsprotokolle für Smartcards

Smart ausweisen

Sichere Authentifizierung verlangt nach starker Kryptographie. Wer auf Passwörter verzichten und sie durch lange Schlüssel ersetzen will, braucht Hilfe beim Rechnen – am besten von einer Chipkarte. Dieser Artikel erklärt die Grundlagen der Challenge-Response-Protokolle und der Smartcard-APIs. Michael Pramateftakis



Bei der Authentifizierung (Griechisch: Authentikotita, die Echtheit) prüft eine Instanz die Identität einer anderen. Sicher zu wissen, wer sich eingeloggt hat, ist entscheidend für die Rechnersicherheit: Das System muss die Identität des Users eindeutig verifizieren, bevor es ihm Zugriff auf Dateien und Programme gewährt. Entsprechend hoch wäre der Schaden, wenn ein Angreifer eine fremde Identität vortäuschen könnte.

Es gibt drei Varianten, wie eine Person ihre Identität beweisen kann: persönliche Merkmale, Besitz und Wissen. Zu den persönlichen Merkmalen gehören das Aussehen, die Stimme oder der Fingerabdruck, also um so genannte biometrische Merkmale.

Bei der zweiten Variante muss die Person ein bestimmtes Objekt besitzen, beispielsweise einen Ausweis. Dieses Objekt muss eindeutig als authentisch erkannt werden und eindeutig zu dieser Person gehören. Beim Personalausweis ist beides gegeben: Das Foto belegen, dass er zu der Person gehört, und besondere Maßnahmen bei der Herstellung gewährleisten, dass sich gefälschte Ausweise von echten unterscheiden.

Das dritte sichere Merkmal ist geheimes Wissen. In der PC-Welt ist das Geheimnis des Benutzers sein Passwort. Wenn jemand genau dieses Passwort erwähnt, muss er der berechtigte Benutzer sein, da es sonst niemand kennt. Ähnliche Verfahren finden sich in vielen Bereichen – um etwa per Telefon Bankgeschäfte auszuführen, braucht der Kunde ein Passwort.

Beweis der Identität

Die Computertechnik ist nicht auf die dritte Variante eingeschränkt. Ein System kann alle drei Methoden verwenden, also natürliche Merkmale, Besitz und geheimes Wissen prüfen. Der Besitz ist zum Beispiel entscheidend, wenn jemand mit seiner Magnetstreifenkarte einen Türöffner aktiviert. Die bekanntesten biometrischen Systeme sind Fingerabdruck-Leser. Es gibt auch Ansätze für Gesichts- und Stimmenerkennung, ebenso zur Erkennung der Handgeometrie oder der Irismuster im Auge. Diese Technologien sind jedoch relativ neu, nicht immer ausgereift und datenschutztechnisch umstritten.

Bei der Authentifizierung mit Passwort gibt jeder Benutzer sein Geheimnis preis, er tippt es ein. Um es geheim zu halten, muss die Umgebung sicher sein, sodass kein Angreifer das Passwort abhören kann. In der Realität ist diese Bedingung leider nicht immer gegeben. Wer zu Hause am PC seine Daten eintippt, darf sich zwar noch recht sicher fühlen. Das endet spätestens, wenn er sich mit FTP an einem entfernten Server einloggt. Das Passwort wird im Klartext übertragen und lässt sich während der Übertragung abhören.

Die Herausforderung

Es gibt aber Methoden für eine passwortbasierte Authentifizierung, bei der niemand sein Passwort preisgeben muss. Ein Kandidat muss nur nachweisen, dass er das Geheimnis kennt, er muss es dem Prüfer nicht mitteilen. Eine dieser Methoden ist das Challenge-Response-Verfahren. Mit diesem einfachen Authentifizierungsprotokoll prüft eine Instanz A, ob B ein Geheimnis kennt, ohne dass B dieses Geheimnis übermittelt. Das Geheimnis darf dazu nur A und B bekannt sein.

Nach dem erfolgreichen Ablauf des Protokolls, wie in Abbildung 1 gezeigt, ist sich A über die Identität von B sicher. Das gemeinsame Geheimnis von A und B ist hier der Schlüssel k für einen bestimmten Verschlüsselungsalgorithmus. A möchte prüfen, ob B diesen Schlüssel tatsächlich kennt.

A erzeugt eine Zufallszahl c und sendet sie an B als so genannte Challenge (Herausforderung). B verschlüsselt diese Zufallszahl mit dem geheimen Schlüssel k und erzeugt so die Response r, die Ant-

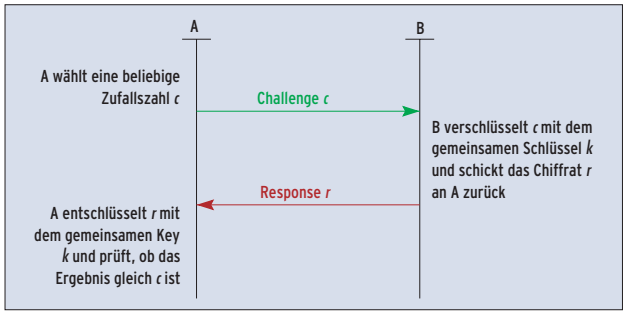


Abbildung 1: Beim Challenge-Response-Protokoll prüft A, ob B ein gemeinsames Geheimnis *k* kennt, ohne dass B dieses Geheimnis preisgibt.

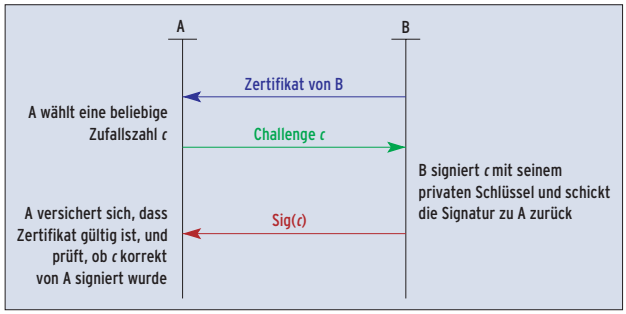


Abbildung 2: Das Challenge-Response-Protokoll mit digitaler Signatur verwendet Zertifikate. A muss den öffentlichen Schlüssel von B vorher nicht kennen.

wort auf die Herausforderung. B schickt diese zurück an A, A entschlüsselt sie mit dem geheimen *k*. A prüft nun, ob die entschlüsselte Nachricht gleich der Zufallszahl *c* ist. Falls ja, hat B denselben Schlüssel zum Verschlüsseln verwendet wie A zum Entschlüsseln. B besitzt also den Schlüssel, seine Identität ist bestätigt. Falls nicht, konnte die andere Seite nicht beweisen, dass sie B ist.

Ein Angreifer, der am Kommunikationskanal zwischen A und B mithört, kann zwar Challenge und Response abfangen, aber nicht den Schlüssel, da er nicht übertragen wird. Aus *c* und *r* kann der Angreifer nicht auf den Schlüssel schließen, da dies voraussetzen würde, dass ihm eine erfolgreiche Known Plaintext Attack auf den Verschlüsselungsalgorithmus möglich ist (siehe Kasten „Angriffe auf Verschlüsselungsalgorithmen"). Solche Angriffe sind auf gute Verschlüsselungsalgorithmen aber praktisch unmöglich. Der Angreifer beobachtet dabei den Kommunikationskanal und legt eine Liste mit Challenges und passenden Antworten an. Später versucht er sich als B auszugeben. Falls eine bekannte Challenge von A kommt, hat der Angreifer die passende Antwort in seiner Liste und kann das Protokoll erfolgreich abschließen. Dieser Fall darf nicht eintreten.

Achtung, Falle

Die Zufallszahlen, die A generiert, müssen also kryptographisch sicher sein. Eine Zufallszahl, die einmal benutzt wurde, darf nie wieder auftreten und niemand darf auf die nächste Zufallszahl schließen können, wenn er die aktuelle Zahl kennt. Eine zusätzliche Sicherung bringen Zeitstempel, die A zusammen mit *c* überträgt und die B mit verschlüsselt. Damit kann sich A versichern, dass die Antwort frisch ist.

Weitere Gefahr droht vom so genannten Man-in-the-Middle-Angriff: Wer Datenverkehr nicht nur abhört, sondern manipuliert, kann das einfache Protokoll missbrauchen. Der Angreifer hört die Challenge, sendet sie an den echten Benutzer, wartet auf dessen Antwort und gibt sie als seine eigene Response aus. Gute Kryptoprotokolle verhindern dies, sind daher aber recht komplex [5]. Beim lokalen Einsatz einer Chipkarte ist dieser Angriff jedoch meist unmöglich.

Wenn nur eine Seite das Protokoll durchführt, spricht man von einseitiger Authentifizierung. Im obigen Beispiel verifiziert A die Identität von B. B muss aber glauben, dass A tatsächlich A ist. Es gibt auch eine Variante für beidseitige Authentifizierung, bei der das Protokoll im Prinzip zweimal abläuft. Beide Runden sind ineinander verschachtelt.

Die Rolle der Chipkarte

Der Nachteil der Challenge-Response-Authentifizierung: Sie ist zu komplex, um sie im Kopf zu berechnen. Der Benutzer braucht elektronische Hilfe, am besten von einer Chipkarte. Die gibt es in verschiedenen Ausführungen, aber nur Prozessorkarten (Smartcards) eignen sich für diese Anwendung. Sie führen kryptographische Operationen durch und speichern Schlüssel sicher.

Der Schlüssel verlässt die sichere Umgebung der Karte dabei nicht. Der Benutzer muss nur eine PIN eingeben (Persönliche Identifikationsnummer), damit ihm die Karte den Zugriff auf den Schlüssel erlaubt. Er muss sich also gegenüber der Karte durch geheimes Wissen authentifizieren. Je nach Kartenleser gibt er die PIN direkt am Kartenleser ein oder er verwendet die PC-Tastatur. Die Logon-

Angriffe auf Verschlüsselungsalgorithmen

Angriffe auf Verschlüsselungsalgorithmen haben das Ziel, den Schlüssel zu ermitteln. Es wird dabei angenommen, dass der Angreifer den Verschlüsselungsalgorithmus kennt. Je nachdem, welche zusätzlichen Informationen ihm zur Verfügung stehen, sind fünf Angriffsarten zu unterscheiden. Gute Algorithmen sind immun gegen die ersten vier.

Ciphertext-only Attack: Der Normalfall. Der Angreifer kennt nur einen Chiffretext und versucht daraus auf den Klartext und den Schlüssel zu schließen.

Known Plaintext Attack: Der Angreifer kennt einen Chiffretext und den zugehörigen Klartext. Aus der Korrelation beider versucht er den verwendeten Schlüssel zu finden.

Chosen Plaintext Attack: Der Angreifer kann beliebige Texte verschlüsseln lassen, kennt aber den Schlüssel nicht. Er wählt spezielle Klartexte, beispielsweise nur Null-Bits, und versucht über Muster im Chiffretext den Schlüssel zu bestimmen.

Chosen Ciphertext Attack: Wie oben, nur umgekehrt. Der Angreifer kann beliebige Chiffretexte entschlüsseln lassen und versucht auf den Schlüssel zu schließen, indem er spezielle Chiffretexte wählt.

Brute Force Attack: Dieses Verfahren ist das einzige, das theoretisch immer funktioniert. Der Angreifer probiert alle Schlüssel durch. Praktisch durchführbar ist dieser Angriff bei modernen Algorithmen jedoch nicht, die Anzahl der möglichen Schlüssel ist zu groß. Für den IDEA-Algorithmus mit 128-Bit-Schlüssel lässt sich folgende Rechnung aufstellen: Wenn alle Menschen auf der Erde zusammen dieselbe Chiffre knacken wollen und jeder Mensch pro Nanosekunde einen Schlüssel prüfen könnte, würde der Brute-Force-Angriff länger dauern als das Universum alt ist.

Umgebung des Betriebssystems muss allerdings auf diese Authentifizierung vorbereitet sein.

Das Protokoll ist sehr einfach, der Aufwand hält sich daher in Grenzen. Wer unter Linux auf Pluggable Authentication Modules setzt (siehe Artikel auf Seite 38), findet beim Muscle-Projekt [12] und bei SCEZ [10] Code für verschiedene Leser und Karten. Auch die Authentifizierungen an entfernten Rechnern ist mit Smartcards effizient und sicher durchzuführen – vorausgesetzt das Protokoll unterstützt dieses Verfahren.

Für Benutzer bestehen sogar Komfortvorteile: Eine PIN ist meist kürzer und leichter zu merken als ein gutes Passwort. Es ist auch möglich, den Desktop automatisch zu sperren, sobald der User seine Karte aus dem Leser entfernt.

Sitzungsschlüssel

Manchmal wird nicht der gespeicherte Schlüssel direkt zum Verschlüsseln der Challenge verwendet, sondern ein davon abgeleiteter Sitzungsschlüssel (Session Key). In diesem Fall müssen beide Instanzen die Herleitungsmethode kennen. Sinnvoll ist es beispielsweise, Kartendaten (etwa die Seriennummer) mit dem Hauptschlüssel zu verknüpfen. Damit kann der Prüfer auch die Chipkarte eindeutig identifizieren.

Die Challenge-Response-Authentifizierung mit Chipkarte behebt viele Probleme der Passwörter. Es gibt keine Möglichkeit mehr, ein schlechtes Passwort zu wählen, alle Schlüssel sind gleich gut. Die Benutzer schreiben ihre Passwörter nicht mehr auf Zettel, die sie an den Monitor kleben. Das Problem kann bei PINs aber erneut auftauchen. Da eine Karte viele Schlüssel speichern kann, ist es bei einem klugen Systemdesign möglich, die Schlüssel für alle Dienste auf der Karte abzulegen und das Problem so einzudämmen. Doch setzt dies voraus, dass alle Dienste passend konfiguriert sind und dasselbe Authentifizierungsverfahren nutzen.

Beidseitige Authentifizierung hätte zudem Vorteile bei öffentlichen Terminals: Die Chipkarte des Benutzers verifiziert zuerst, dass das Terminal authentisch ist, und setzt erst dann die Benutzerauthentifikation fort. Der Benutzer weiß danach, dass er ein authentisches Terminal vor sich hat.

Authentifizierung mit digitalen Signaturen

Das Challenge-Response-Protokoll lässt sich auch mit einer digitalen Signatur statt der Verschlüsselung betreiben. Die Signatur ersetzt die Verschlüsselungsoperation der Challenge. Statt die Response zu entschlüsseln, verifiziert der Prüfer die Signatur. Kann A die Signatur erfolgreich prüfen, ist die Identität von B bewiesen. Der Vorgang ist in Abbildung 2 dargestellt, Näheres steht im Kasten „Grundlagen der Kryptographie“.

Die Implementierung verlangt eine Chipkarte, die in der Lage ist, digitale Signaturen zu berechnen. Das ist meist nur ein Kostenproblem – die Karten kosten das Mehrfache einer einfachen Ausführung. Das Verfahren spart dafür Konfigurationsaufwand: Das System muss nur die Namen der Benutzer und ihre Rechte kennen. Die geheimen Schlüssel dürfen nicht bekannt sein, nicht einmal die öffentlichen Schlüssel der Benutzer müssen im Computer gespeichert sein.

Die Zuordnung von Benutzerkennung zu öffentlichem Schlüssel übernimmt ein Zertifikat, das die Karte vor der Challenge-Response-Runde übermittelt. Dazu ist allerdings eine PKI nötig [4], die einen zusätzlichen Kosten- und Komplexitätsfaktor in das System bringt. Ein Authentifizierungssystem mit digitalen Signaturen ist daher nur in Umgebungen mit vielen Benutzern vorteilhaft.

Ein Nachteil der Kryptoverfahren ist die Interoperabilität. Die Benutzer können zwar im eigenen Netzwerk einheitlich ohne Passwort arbeiten. Sobald sie externe Dienste beanspruchen, brauchen sie aber wieder Passwörter, da fremde Dienste meist nicht auf Chipkarten-Authentifikation vorbereitet sind.

Um die Challenge-Response-Verfahren selbst zu implementieren, sind eine Chipkarte und ein Leser nötig. Die Smartcard sollte idealerweise asymmetrische Kryptographie beherrschen und der Kartenleser braucht einen passenden Treiber. Dann fehlt nur noch das Authen-

Grundlagen der Kryptographie

Die Kryptographie unterscheidet symmetrische und asymmetrische Algorithmen. Die symmetrischen verwenden zum Chiffrieren und Dechiffrieren denselben Schlüssel. Beide Kommunikationspartner müssen diesen geheimen Key kennen. Ein wesentliches Problem ist folglich der Schlüsselaustausch.

Asymmetrische Kryptographie löst dies, indem sie pro Kommunikationspartner zwei Schlüssel verwendet – einen öffentlichen und einen dazu passenden privaten Schlüssel. Der private Key muss geheim bleiben. Nur der öffentliche Schlüssel muss dem Partner bekannt sein. Die Übertragung ist einfacher als bei der symmetrischen Kryptographie, da der öffentliche Schlüssel nicht geheim bleiben muss.

Jede Krypto-Operation mit einem der Schlüssel ist nur mit dem passenden Gegenstück rückgängig zu machen. Eine chiffrierte Nachricht muss mit dem öffentlichen Key des Empfängers verschlüsselt sein. Sie lässt sich dann nur mit dessen privatem Schlüssel dechiffrieren.

Verschlüsseln und signieren

Der Absender kann eine Nachricht auch mit seinem privaten Key verschlüsseln. Dann ist jeder in der Lage, mit dem zugehörigen öffentlichen Schlüssel zu entschlüsseln. Das ist die Basis der digitalen Signatur: Die Nachricht muss nicht geheim bleiben, der Empfänger möchte nur ihren Ursprung kennen. Da nur der Absender den privaten Schlüssel kennt, kann er als einzige Person eine gültige Signatur erstellen. Jeder kann aber die Signatur mit dem öffentlichen Schlüssel des Signierers verifizieren.

Um eine digitale Signatur zu prüfen, muss der Empfänger sicher sein, dass er den richtigen öffentlichen Schlüssel kennt. Nur so kann er von der Signatur auf die Identität des Signierers schließen. Die Zuordnung eines öffentlichen Schlüssels zu einer Person lässt sich am besten durch ein Zertifikat beweisen: Es enthält den öffentlichen Schlüssel sowie die Bezeichnung der Person. Ist ein Zertifikat von einer vertrauenswürdigen Instanz signiert, kann der Empfänger der Zuordnung vertrauen.

Heute gibt es viele Zertifizierungsstellen. Es ist daher sehr wahrscheinlich, dass ein Empfänger Zertifikate zu verifizieren hat, die nicht von einer ihm bekannten Zertifizierungsstelle stammen. Es müssen also Mechanismen vorhanden sein, die es ihm trotzdem erlauben: die PKI (Public Key Infrastructure, [4]). Wenn die Zertifikate sogar vor Gericht gültig sein sollen, ist der Aufbau einer PKI sehr aufwändig, da der Gesetzgeber hohe Sicherheitsanforderungen an die Zertifizierungsstelle stellt.

tifizierungsprotokoll, das es in die gewünschte Software einzubinden gilt. Unter Linux bietet es sich an, ein PAM-Modul zu entwickeln (Artikel auf Seite 38).

Die Wahl der Karte

Auf dem Markt sind viele geeignete Chipkarten vertreten. Jeder Hersteller bietet Modelle für verschiedene Aufgabengebiete an. Die Wahl hängt von mehreren Faktoren ab: Kosten, benötigte Anzahl, Erweiterbarkeit um andere Applikationen, Flexibilität und vieles mehr. Karten, die digital signieren, sind deutlich teurer als Karten, die nur symmetrische Algorithmen ausführen. Auch sind Karten, die mehrere Applikationen aufnehmen, teurer als solche, die nur eine Applikation ausführen.

In ihrer Flexibilität unterscheiden sich frei programmierbare Karten, etwa Java- und Basic-Karten, von solchen, die nur den ISO-Befehlssatz kennen. Beispiele für programmierbare Highend-Java-Karten sind die Smartcafé-Karte von Giesecke und Devrient **[6]** und die Gemxpresso-Karte von Gemplus **[7]**. Unter den ISO-Karten hat sich beispielsweise die Starcos-Karte von Giesecke und Devrient bewährt. Wer eine günstige, frei programmierbare Karte sucht, aber auf die Multiapplikationsfähigkeit verzichten kann, wird bei der Basiccard von Zeitcontrol **[8]** fündig.

Fast alle Chipkartenhersteller bieten auch verschiedene Kartenleser an **[3]**. Die Geräte lassen sich in drei Klassen einteilen: die reinen Kartenleser (auch Klasse-1-Leser genannt), Kartenleser mit eigener PIN-Tastatur (Klasse 2) und Kartenleser mit PIN-Tastatur und Display (Klasse 3). Programmierer haben die Wahl zwischen mehreren Möglichkeiten, um auf Kartenleser und Karten zuzugreifen. Es gibt APIs für C/C++ (etwa PC/SC und CT-API) sowie für Java (das Opencard Framework).

PC/SC **[11]** ist der zurzeit wichtigste Standard, er wird auch von den meisten Kartenlesern unterstützt. Unter Linux bietet das Open-Source-Projekt Muscle **[12]** eine Implementierung.

Der PC/SC-Standard

PC/SC besteht aus mehreren Schichten, wie in Abbildung 3 zu sehen ist. Ganz unten befindet sich der Kartenleser mit der Karte. Der Treiber des Kartenlesers (Interface Device Handler) kommuniziert mit dem PC/SC Resource Manager, der den Zugriff auf den Leser und die Karte regelt.

So ist es möglich, dass mehrere Programme gleichzeitig auf eine bestimmte Karte zugreifen und ein Programm mehrere Leser benutzt. Die Applikationen können direkt über den Resource Manager mit den Karten kommunizieren oder über einen Service Provider, der komplexere Funktionalität bereitstellt. ▶

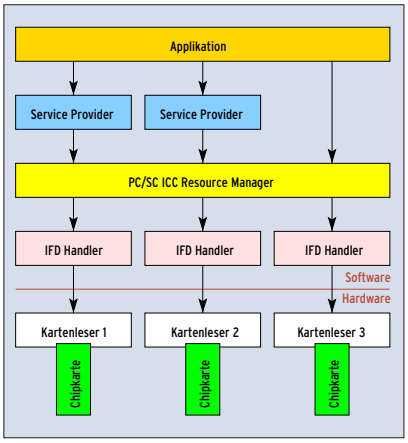


Abbildung 3: Bei PC/SC greift die Applikation über Service Provider auf den ICC Resource Manager zu (IC-Card, Chipkarte). Der Kartenleser ist per IFD-Handler angeschlossen (Interface Device, Leser).

Um die Karte anzusprechen, muss eigene Software zuerst eine Verbindung mit dem Resource Manager über den Aufruf »SCardEstablishContext()« herstellen. Die Funktion »SCardConnect()« bindet den Kontext an einen Kartenleser. »SCardTransmit()« schickt einen Befehl zur Karte und liest die Antwort. »SCardDisconnect()« trennt die Verbindung zur Karte, »SCardReleaseContext()« trennt die Verbindung zum Resource Manager. Eine detaillierte Beschreibung und Beispielcode ist in der PC/SC-Dokumentation zu finden. Eine Übersicht gibt [2].

CT-API: Drei Funktionen genügen

Der CT-API-Standard ist in Deutschland verbreitet. Er bietet das einfachste Interface für Chipkarten. Drei Funktionen genügen: »CT_init()« baut eine Verbindung mit dem Kartenleser auf, »CT_data()« sendet einen Befehl zur Karte oder zum Leser und »CT_close()« trennt die Verbindung mit dem Kartenleser.

Mit CT-API kann ein Programm Befehle auch direkt zum Kartenleser senden, indem es eine bestimmte Zieladresse beim »CT_data()«-Aufruf angibt. Ein zusätzlicher Standard (CT-BCS, Basic Command Set) definiert Kartenleserbefehle. Damit kann die Software auf eine Karte warten, den Status des Lesers abfragen oder Tastatur und Display der Klasse-2- und Klasse-3-Leser nutzen.

Für den Kartenleserzugriff mit Java gibt es das Opencard Framework [9]. OCF verwendet eine ähnliche Architektur wie PC/SC, siehe Abbildung 4. Jeder Kartenleser wird von einer »CardTerminal«-Klasse angesteuert, die den Treiber des Kartenlesers für die Java-Welt darstellt. Diese Klasse verwendet meist das JNI (Java Native Interface), um einen Low-Level-Treiber für den Kartenleser anzusprechen. Es gibt aber auch so genannte Pure-Java-Terminals, die ohne native Komponenten die Leser ansteuern (beispielsweise über Java-Comm).

Opencard für Java-Programmierer

Der Zugriff auf die Karten erfolgt über »CardService«-Klassen, die komplexere Abläufe kapseln. Es gibt zum Beispiel eine Klasse »FileAccessCardService«, die einen einfachen Zugriff auf das Dateisystem einer ISO-Karte ermöglicht. Die Klasse »PassThruCardService« ist die einfachste »CardService«-Klasse und bietet transparenten Zugriff auf die Karte. Der Programmierer muss dann selbst die Chipkartenbefehle erzeugen.

Zur Wahl stehen auch spezialisierte Bibliotheken wie SCEZ [10]. Diese Libraries unterstützen meist nur bestimmte Leser und Karten und sind für einfache Applikationen gedacht, können also nicht mit der Flexibilität von PC/SC oder OCF mithalten. Jeder Entwickler muss selbst entscheiden, welches API sich für sein Projekt am besten eignet.

ISO-Karten bieten spezielle Befehle für einseitige und zweiseitige Authentifikation mit symmetrischen Schlüsseln, der Programmierer braucht sich nur um die PC-seitige Implementierung zu kümmern. Falls er dagegen eine programmierbare Karte einsetzt, muss er zusätzlich zum PC-Programm ein entsprechendes Programm für die Karte entwickeln.

Gute Alternative

Die Authentifizierung mit Chipkarte und Challenge-Response-Verfahren ist eine gute Alternative zum Passwortsystem. Ob mit symmetrischen Schlüsseln oder mit digitalen Signaturen und Zertifikaten: Es bietet ein hohes Sicherheitsniveau und einige Vorteile für den Benutzer. Derzeit sind noch Modifikationen am Betriebssystem nötig. Mit der zunehmenden Verbreitung von Chipkarten werden die Betriebssysteme diese Technik künftig direkt unterstützen. *(fjl)* ■

Infos

[1] Frank Haubenschild, „Smartcard-Einsatz unter Linux“: Linux-Magazin 05/02, S. 90
[2] Frank Haubenschild, „Smartcard-Programmierung“: Linux-Magazin 06/02, S. 98
[3] Andreas Schmoll und Mirko Dölle, „Smartcard-Reader mit verschiedenen Schnittstellen“: Linux-Magazin 08/03, S. 56
[4] Kay Wondollek, „Vertrauens-Frage: Public Key Infrastructure – Architektur und Software“: Linux-Magazin 02/04, S. 66
[5] Peter Gutmann, „Schutz (be)dürftig – Fehler in VPN-Protokollen und deren Lösung“: Linux-Magazin 01/04, S. 84
[6] Giesecke & Devrient: [http://www.gdm.de]
[7] Gemplus: [http://www.gemplus.com]
[8] Zeitcontrol: [http://www.zeitcontrol.de]
[9] Opencard Framework: [http://www.opencard.org]
[10] SCEZ Smartcard Library: [http://www.franken.de/crypt/scez.html]
[11] PC/SC-Workgroup: [http://www.pcscworkgroup.com]
[12] Muscle: [http://www.linuxnet.com]

Der Autor

Michael Pramateftakis ist wissenschaftlicher Mitarbeiter am Lehrstuhl für Datenverarbeitung, Technischen Universität München. Sein Arbeitsgebiet ist Systemsicherheit und Kryptologie mit Schwerpunkt Chipkarten und E-Commerce.

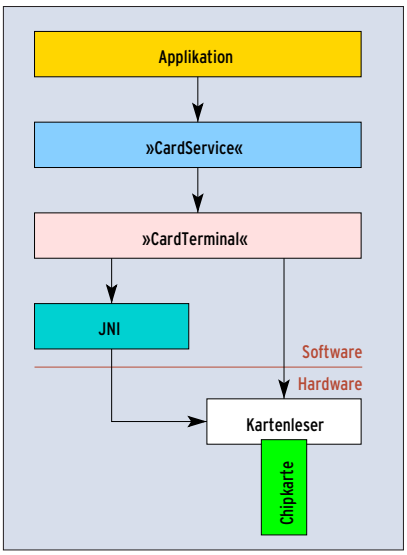


Abbildung 4: In Opencard abstrahiert »CardTerminal« den Kartenleser, »CardService« bietet einen einheitlichen Zugang zu den Diensten der Chipkarte.