

Smartcards unter Linux

Praktische Anwendungsfälle

Dominik Fischer



2016-01-09

- 1 Grundlagen
- 2 SSH
- 3 Login mit PAM
- 4 GnuPG
- 5 Festplattenverschlüsselung
- 6 Links

Grundlagen

Smartcards (*ICC – integrated circuit card*) gibt es vielen Ausführungen. Sie unterscheiden sich in folgenden Punkten:

- ▶ Format: ID-1 (EC-Karte), ID-000 (normale SIM-Karten), ...
- ▶ kontaktbehaftet / kontaktlos
- ▶ Betriebssysteme



- ▶ ROM (Betriebssystem), RAM (Arbeitsdaten), EEPROM (Dateisystem)
- ▶ Dateisystem (standardisiert in ISO 7816-4)
 - ▶ MF (master file): Root-Verzeichnis
 - ▶ DF (dedicated file): Verzeichnisdateien
 - ▶ EF (elementary file): Nutzdaten
 - ▶ Inhalt durchstöbern z.B. mit: `opensc-browser`

Smartcardleser (*IFD – interface device*) werden gemäß ihrer Sicherheit in drei Klassen unterteilt.

- ▶ Class-1: Nur Kontakteinheit
- ▶ Class-2: Kontakteinheit mit Tastatur zur PIN-Eingabe
- ▶ Class-3: Kontakteinheit mit Tastatur und Display



Benötigte Softwarekomponenten installieren:

- ▶ Gerätetreiber
- ▶ API für Low-Level Zugriff:
 - ▶ OCF (open card framework)
 - ▶ PC/SC (personal computer/smart card)
- ▶ Crypto-API

```
$ sudo apt-get install \
    libccid pcscd \
    opensc opensc-pkcs11
```

- ▶ Erkannte Reader auflisten:

```
# opensc-tool -l
# Detected readers (pcsc)
Nr. Card Features Name
0 Yes Gemalto GemPC Express(...)
```

- ▶ ATR der Karte auflisten

```
# opensc-tool -r 0 -a
3b:fe:18:00:00:81:31:fe:45:80:31:81:54:48(...)
```

- ▶ Karte identifizieren

```
# opensc-tool -r 0 -n
SmartCard-HSM
```

- ▶ Dateien auf der Karte auflisten

```
# opensc-tool -r 0 -f
3f00 type: DF, size: 0
(...)
```

Normalerweise wird `pkcs15-init` benutzt, aber bei dem hier verwendeten Smartcard-Typ funktioniert das nicht. Stattdessen muss das Tool `sc-hsm-tool` verwendet werden.

- ▶ SO-PIN: genau 16 hexadezimale Ziffern
- ▶ User-Pin: 6 bis 16 ASCII-Zeichen, kann geändert werden, aber nicht die Länge (dafür muss man reinitialisieren und alles Löschen)

```
# sc-hsm-tool --initialize \
--so-pin 0123012301230123 \
--pin 123456 \
--label Chaosconsulting
```

- ▶ Objekte der Karte dumpen

```
# pkcs15-tool -D
Using reader with a card: Gemalto GemPC Express 00 00
PKCS#15 Card [Chaosconsulting]:
  Version        : 0
  Serial number  : DECM0102470
  Manufacturer ID: www.CardContact.de
  Flags          : PRN generation

PIN [UserPIN]
  Object Flags   : [0x3], private, modifiable
  (...)

PIN [SOPIN]
  Object Flags   : [0x1], private
  (...)
```

Schlüsselpaar erzeugen: pkcs11-tool



```
# pkcs11-tool -l --keypairgen \
    --key-type rsa:2048 \
    --id 10 \
    --label "My first RSA keypair"

Using slot 1 with a present token (0x1)
Logging in to "Chaosconsulting (UserPIN)".
Please enter User PIN:
Key pair generated:
Private Key Object; RSA
    label:      My first RSA keypair
    ID:        10
    Usage:     decrypt, sign, unwrap
Public Key Object; RSA 2048 bits
    label:      My first RSA keypair
    ID:        10
    Usage:     encrypt, verify, wrap
```

- ▶ *PublicKey* auslesen

```
$ pkcs15-tool --read-public-key 10 > pubkey.pem
$ echo "Hallo Welt!" > foobar.txt
$ sha256sum foobar.txt
1cf0c75265e678c4b9bcc32a798e8d484b (...)
```

- ▶ Datei mit *PublicKey* verschlüsseln

```
$ openssl rsautl -inkey pubkey.pem \
    -pubin -encrypt -pkcs -in foobar.txt \
    -out foobar.txt.encrypted
$ sha256sum foobar.txt.encrypted
fbc38a86a60c500aab1251ec1110d41ebe (...)
```

- ▶ Datei mit *PrivateKey* von Smartcard entschlüsseln

```
$ pkcs15-crypt --decipher --key 10 \
    --input foobar.txt.encrypted \
    --pkcs1 --raw > foobar.txt.encrypted.decrypted
$ sha256sum foobar.txt.encrypted.decrypted
1cf0c75265e678c4b9bcc32a798e8d48 (...)
```

SSH

- ▶ Public-Key im SSH-Format ausgeben

```
$ pkcs15-tool --read-ssh-key 10
Using reader with a card: Gemalto GemPC Express 00 00
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQDIOT1V (...)
```

- ▶ Der *PublicKey* muss in die `~/.ssh/authorized_keys` auf dem entfernten Host eingetragen werden
- ▶ Smartcard mit SSH verwenden. Als Identity-File wird die PKCS#11 Library angegeben.

```
$ ssh -I /usr/lib/x86_64-linux-gnu/opensc-pkcs11.so \
      user@entfernter.host
Enter PIN for 'Chaosconsulting (UserPIN)':
```

- ▶ Der `ssh-agent` wird auf den meisten Linux-Systemen automatisch beim Login gestartet
- ▶ Der `gnome-keyring-daemon` darf nicht dazwischen kommen!
- ▶ Token zum SSH-Agenten hinzufügen

```
$ ssh-add -s /usr/lib/x86_64-linux-gnu/opensc-pkcs11.so
```

- ▶ Vom ssh-agent verwaltete Schlüssel anzeigen

```
$ ssh-add -l
```

- ▶ Token vom SSH-Agenten entfernen

```
$ ssh-add -e /usr/lib/x86_64-linux-gnu/opensc-pkcs11.so
```

Login mit PAM

- ▶ Nötige Pakete installieren

```
# apt-get install libpam-pkcs11 libengine-pkcs11-openssl
```

- ▶ Konfiguration /etc/pam_pkcs11/pam_pkcs11.conf anlegen

```
$ cp /usr/share/doc/libpam-pkcs11/examples/pam_pkcs11.conf.example \
      /etc/pam_pkcs11/pam_pkcs11.conf
$ vi /etc/pam_pkcs11/pam_pkcs11.conf
pam_pkcs11 {
    ...
    use_pkcs11_module = opensc;
    pkcs11_module opensc {
        module = /usr/lib/x86_64-linux-gnu/opensc-pkcs11.so;
        ...
    }
    use_mappers = openssh;
    ...
}
```

Für den PAM Typ *auth* muss das pam_pkcs11 Modul bei den gewünschten Services eingetragen werden.

Z.B. für Logins in der Datei /etc/pam.d/sudo:

```
auth      sufficient      pam_pkcs11.so nullok try_first_pass  
(...)
```

Ähnliche Einträge werden bei den anderen Diensten vorgenommen (z.B. für das grafische Login, sudo, screensaver, etc.).

!!! Vorsicht bei der Anpassung von PAM !!!

Vorher immer mehrere Logins mit Root-Sessions öffnen und gut testen, damit man sich nicht aussperrt!!!

- ▶ Nötige Pakete installieren

```
\# apt-get install libpam-pkcs11 libengine-pkcs11-openssl
```

- ▶ Zertifikat erstellen

```
$ openssl
# Engine einrichten...
OpenSSL> engine -t dynamic \
    -pre SO_PATH:/usr/lib/engines/engine_pkcs11.so
    -pre ID:pkcs11 -pre LIST_ADD:1 -pre LOAD \
    -pre MODULE_PATH:/usr/lib/x86_64-linux-gnu/openssl-pkcs11.so

# Certificate Signing Request erstellen...
OpenSSL> req -engine pkcs11 -new -key 1:10 \
    -keyform engine -out req.pem
    -text -x509 -subj "/CN=Dominik Fischer"

# Zertifikat ausstellen...
OpenSSL> x509 -engine pkcs11 -signkey 1:10 \
    -keyform engine -in req.pem -out cert.pem
```

- ▶ Dateiformat umwandeln

```
$ openssl x509 -in cert.pem -out cert.der -outform der
```

- ▶ Zertifikat auf Smartcard schreiben

```
$ pkcs11-tool \  
  --module /usr/lib/x86_64-linux-gnu/opensc-pkcs11.so \  
  -l --write-object cert.pem \  
  --type cert \  
  --id 45 --label "MyCert"
```

- ▶ Da wir den openssh Mapper beim pam_pkcs11 ausgewählt haben, Wie zuvor bei SSH: Key im SSH Format auslesen und in die `authorized_keys` des Accounts eintragen.

GnuPG

- ▶ Die OpenPGP ist ähnlich wie die bisher besprochenen Karten. Sie hat eine vordefinierte Datenstruktur (`pkcs15-tool -D` zeigt sie an).
- ▶ GnuPG hat Optionen zur Verwaltung der Karte
- ▶ Kartenstatus anzeigen:

```
$ gpg --card-status
gpg: detected reader 'Gemalto GemPC Express 01 00'
Application ID ....: D2760001240102010005000041BC0000
Version .....: 2.1
Manufacturer .....: ZeitControl
(...)
```

- ▶ Karteninhalt editieren:

```
$ gpg --card-edit
(...)
gpg/card> help
quit      Men verlassen
admin     Zeige Admin-Befehle
(``')
```

- ▶ Verschlüsseln

```
$ gpg -ea -r mmuster@example.com foobar.txt
```

- ▶ Entschlüsseln

```
$ gpg -d --output foobar.txt.asc.decrypted foobar.txt.asc
```

- ▶ Public Key muss außerhalb der Karte gespeichert sein. Am einfachsten direkt nach der Erstellung auf einen Keyserver hochladen:

```
$ gpg --send-keys 6C7F8E30
```

- ▶ Auf frischem Rechner ausführen:

```
$ gpg --list-keys
$ gpg --card-edit
gpg/card> fetch
```

Damit wurden die nötigen Dateien unter `~/.gnupg` erstellt.

Festplattenverschlüsselung

- ▶ Die Verwendung von LUKS mit Smartcard ist möglich, wird hier allerdings nicht betrachtet.
- ▶ Veracrypt (Nachfolger von TrueCrypt) unterstützt PKCS#11-Tokens und damit auch Smartcards.

- ▶ Beim Anlegen des Volumes muss ein Keyfile erstellt werden
- ▶ Das Keyfile wird dann im Terminal per PIN geschützt als Objekt auf die Smartcard übertragen
- ▶ In den Veracrypt Einstellungen muss der Pfad zur opensc-pkcs11 Library eingetragen werden (Settings – Preferences... – Security Token)

Links

- ▶ Wikipedia zu Chipkarten:
<https://de.wikipedia.org/wiki/Chipkarte>
- ▶ OpenPGP-Card: <http://shop.kernelconcepts.de>
- ▶ SmartCard-HSM: <http://www.cardomatic.de/>
- ▶ OpenSC Projekt: <https://github.com/OpenSC>
- ▶ PKCS: https://de.wikipedia.org/wiki/Public-Key_Cryptography_Standards
- ▶ Veracrypt: <https://veracrypt.codeplex.com/>