

15 September 2015

I have learned today that all PGP public keys of John Young <jya@pipeline.com> and Cryptome <cryptome@earthlink.net> have been compromised.

The keys have been revoked today.

Two new keys have been generated today:

John Young 15-0915 <jya@pipeline.com> 0xD87D436C

Cryptome 15-0915 <cryptome@earthlink.net> 0x8CD47BD5

This message is signed by the first.

Hardware Security Module in der Praxis

- Heinlein Support
 - IT-Consulting und 24/7 Linux-Support mit ~28 Mitarbeitern
 - Eigener Betrieb eines ISPs seit 1992
 - Täglich tiefe Einblicke in die Herzen der IT aller Unternehmensgrößen
- 24/7-Notfall-Hotline: 030 / 40 50 5 - 110
 - 28 Spezialisten mit LPIC-2 und LPIC-3
 - Für alles rund um Linux & Server & DMZ
 - Akutes: Downtimes, Performanceprobleme, Hackereinbrüche, Datenverlust
 - Strategisches: Revision, Planung, Beratung, Konfigurationshilfe

Inhaltsübersicht

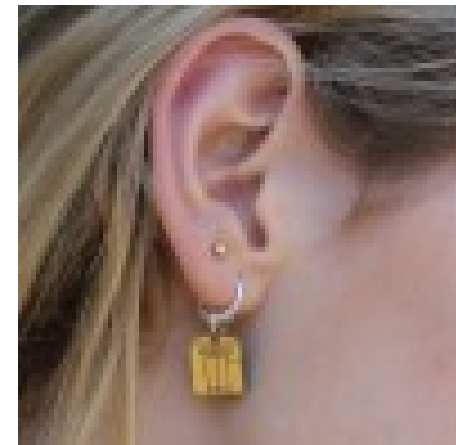
- 1. Zwei-Faktor-Authentifizierung mit OTP- und U2F-Token
- 2. OpenPGP Smartcards in der Praxis mit GnuPG
 - 2.1 Verschlüsseln/Entschlüsseln von E-Mails und Dateien mit OpenPGP
 - 2.2 Sichere Authentifizierung für SSH mit OpenPGP Smartcards
 - 2.3 Authentifizierung bei einem Git Repo mit OpenPGP Smartcards
 - 2.4 StrongSwan Client Authentifizierung mit OpenPGP Smartcards
 - 2.5 Root-CA Zertifikate mit XCA und OpenPGP Smartcards
 - 2.6 OpenPGP Smartcards mit gpgsm verwenden
- 3. Hardware Security Module für X.509 mit OpenSC
 - 3.1 Hardware Security Module initialisieren
 - 3.2 Hardware Security Module mit openssl nutzen
 - 3.3 Hardware Security Module mit Firefox und Thunderbird
 - 3.4 HSMs mit StrongSwan (Client Auth) und PowerDNS (DNSSEC Sig.) nutzen

1. Zwei-Faktor-Authentifizierung

- Kombination von „Wissen“ und „Besitz“ für Authentifizierung
 - Weicher Besitz: Irgendwelche kopierbaren Bits und Bytes auf einem Smartphone o.ä.
 - Harter Besitz: Physikalisch vorhandenes, nicht kopierbares Hardware Token

1. Zwei-Faktor-Authentifizierung

- Kombination von „Wissen“ und „Besitz“ für Authentifizierung
 - Weicher Besitz: Irgendwelche kopierbaren Bits und Bytes auf einem Smartphone o.ä.
 - Harter Besitz: Physikalisch vorhandenes, nicht kopierbares Hardware Token (z.B. Yubikeys oder NitroKey Pro + App)



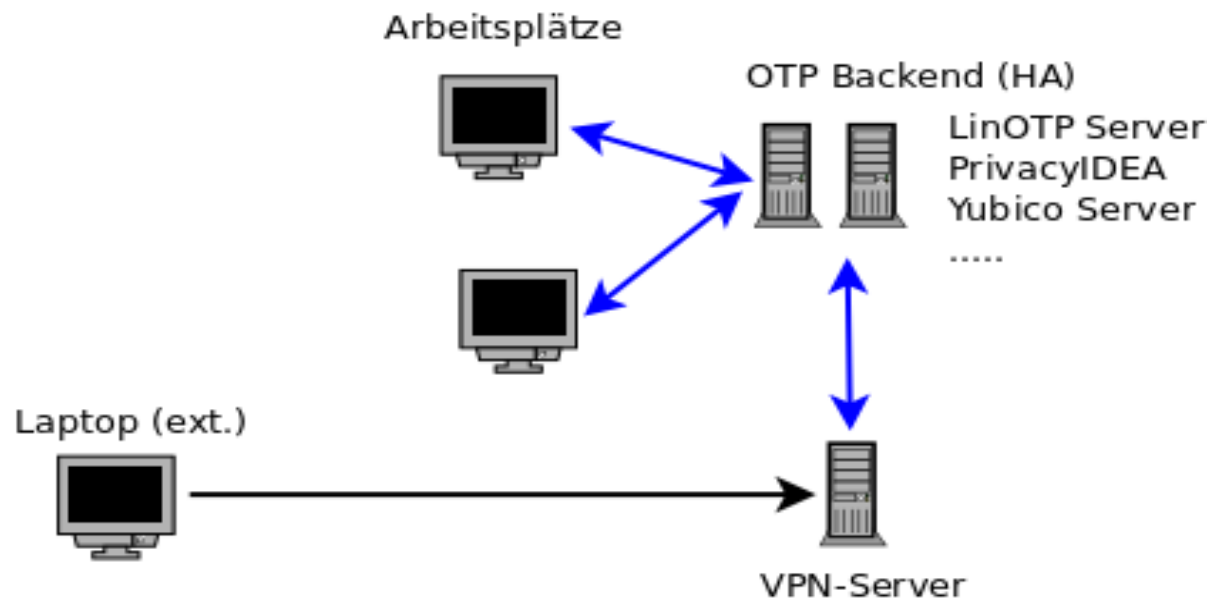
1. Zwei-Faktor-Authentifizierung

- Anwendungsbeispiele:
 - Login für Webdienste (<http://www.dongleauth.com>)
 - Login in Firmennetzen (Desktop Computer, VPN)
 - Zugriff auf verschlüsselte Datenbereiche

1. Zwei-Faktor-Authentifizierung

→ Anwendungsbeispiele:

- Login für Webdienste (<http://www.dongleauth.com>)
- Login in Firmennetzen (Desktop Computer, VPN)
- Zugriff auf verschlüsselte Datenbereiche
- Ein einfaches Beispiel für eine kleine Firma: Login mit PAM-Modulen

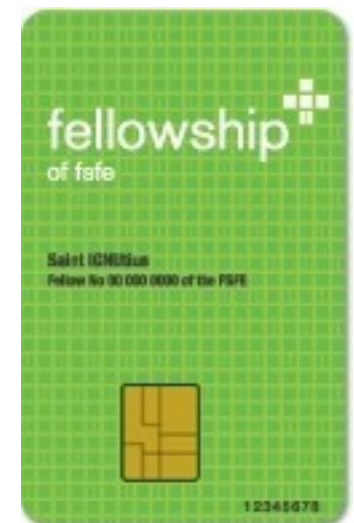


2. OpenPGP Smartcards

- Der private Schlüssel ist auf der Smartcard gespeichert und verläßt diese sichere Umgebung nie, alle Crpyto-Op. laufen auf der Smartcard.

2. OpenPGP Smartcards

- Der private Schlüssel ist auf der Smartcard gespeichert und verläßt diese sichere Umgebung nie, alle Crypto-Op. laufen auf der Smartcard.
- USB-Stick Format
 - Nitrokey (Open Source Hardware Projekt, <https://www.nitrokey.com>)
 - GnuK (Open Source Hardware Projekt der Free Software Foundation Japan)
 - Yubikey (Produkt der Firma Yubico, <https://www.yubico.com>)
- Checkkarten Format
 - Kernel concepts G10 Card
 - FSFE Fellowship Card



2.1 E-Mails verschlüsseln mit Smartcards

Mit der GnuPG Software Kollektion funktionieren Smartcards out-of-the-box (gpg2, gpgsm, gpg-agent, sddaemon)

→ Vorbereitungen:

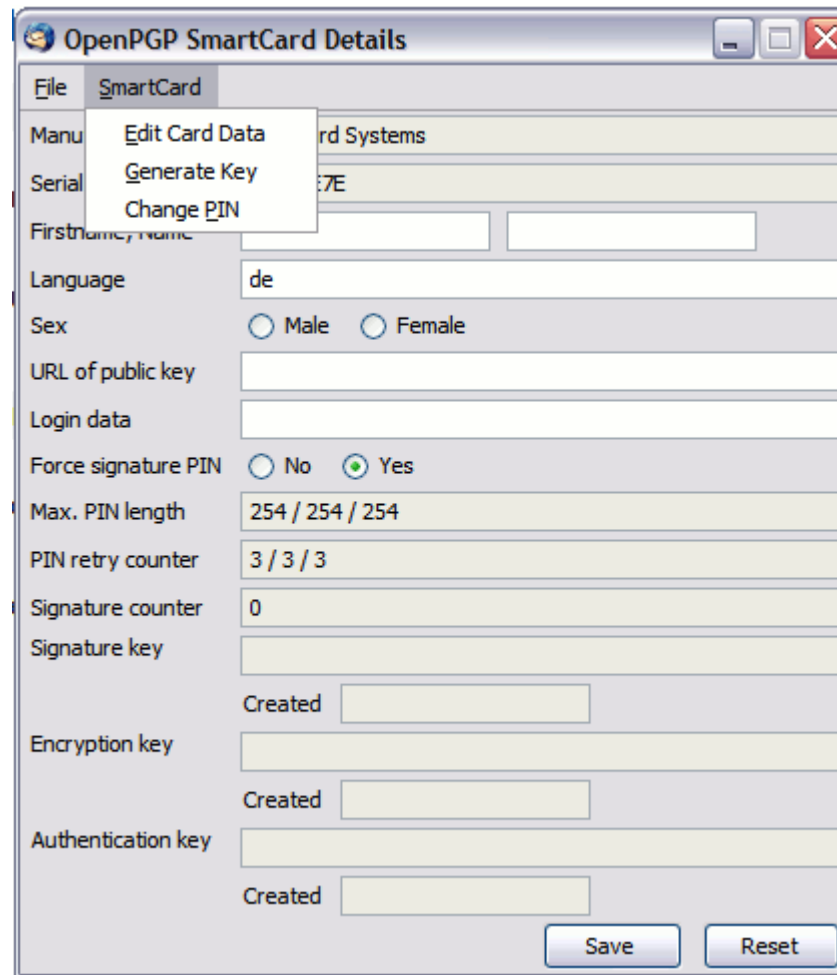
1. OpenPGP-Schlüssel erzeugen (Kommandozeile oder GUI wie Enigmail.)
2. Daten auf der Smartcard anpassen (insb. Download URL für public Key)
3. Default PIN und Admin-PIN ändern (Default: „123456“ und „123456768“)
4. Öffentlichen Schlüssel verteilen und zur Download Location hochladen

2.1 E-Mails verschlüsseln mit Smartcards

```
> gpg2 --card-edit
...
gpg/card> admin
Admin-Befehle sind erlaubt

gpg/card> help
....
gpg/card> quit
>
```

2.1 E-Mails verschlüsseln mit Smartcards



2.1 E-Mails verschlüsseln mit Smartcards

Neuer Computer, wie bekommt man den OpenPGP Key?

- (a) Backup des Schlüsselpaare einspielen (wie üblich)
- (b) Smartcards bieten noch eine andere Möglichkeit:

```
> gpg2 --card-edit
```

```
...
```

```
gpg/card> fetch           (Abrufen des öffentlichen Schlüssel von der URL)
```

```
gpg/card> quit
```

```
> gpg2 --card-status     (Private Schlüssel als Referenz auf Card anlegen)
```

2.1 E-Mails verschlüsseln mit Smartcards

Bekannte Probleme:

→ GNOME keyring manager hijackes the GnuPG agent → No Smartcard found!

Lösung 1: GNOME Keyring Manager deinstallieren

Lösung 2: GNOME Keyring Manager umkonfigurieren

2.1 E-Mails verschlüsseln mit Smartcards

Bekannte Probleme mit OpenPGP Smartcards:

- **GNOME keyring manager hijackes the GnuPG agent** → No Smartcard found!
Lösung 1: GNOME Keyring Manager deinstallieren
Lösung 2: GNOME Keyring Manager umkonfigurieren

- **Java PGP Implementierungen können nicht mit Smartcards umgehen**
 - Kein Zugriff auf den privaten Schlüssel, weil kein Smartcards Interface implementiert ist.
 - Falsche Verwendung der öffentlichen Schlüssel. In der Regel verwenden (fast) alle Java Implementierungen den Authentication Subkey statt des Encryption Subkey zum Verschlüsseln → Entschlüsseln nicht möglich!

- Lösung: Öffentlichen Schlüssel ohne Authentication Subkey verteilen

2.1 E-Mails verschlüsseln mit Smartcards

Schlüssel ohne Authentication Subkey verteilen

1. Backup des öffentlichen und privaten Schlüssel (PGP Schlüsselverwaltung)
2. Authentication Subkey löschen (auf der Kommandozeile mit gpg2):

```
> gpg2 -edit-key max.mustermann@domain.tld
```

```
...
```

```
pub 4096R/0x....  usage: S,C
```

```
sub 4096R/0x....  usage: A
```

```
sub 4096R/0x....  usage: E
```

```
→
```

```
gpg> key 1
```

```
gpg> delkey
```

```
gpg> quit
```

- Öffentlichen Schlüssel exportieren und verteilen (PGP Schlüsselverwaltung)
- Unter 1. erstelltes Backup wieder importieren (PGP Schlüsselverwaltung)

2.2 SSH mit OpenPGP Smartcard

- OpenPGP Smartcard vorbereiten (PIN ändern, Schlüssel generieren usw.)
- „ssh-agent“ muss auf dem Client Desktop abgeschaltet werden
(Debian/Ubuntu: in „/etc/X11/Xsession.options“ auskommentieren)
- „gpg-agent“ muss die Aufgabe des „ssh-agent“ übernehmen
(in „\$HOME/.gnupg/gpg-agent.conf“ `enable-ssh-support` eintragen)
- Smartcard anschließen und öffentlichen Schlüssel als SSH-Key exportieren
 - (a) `> ssh-add -L > my-ssh-card-key.pub`
 - (b) `> gpgkey2ssh 0x12345678 > my-ssh-card-key.pub`
- Öffentlichen Schlüssel auf SSH-Server oder als autorisiert hinterlegen

(Funktioniert auch mit Putty für Windows, aber wir sind hier auf der SLAC.)

2.2 SSH mit OpenPGP Smartcard

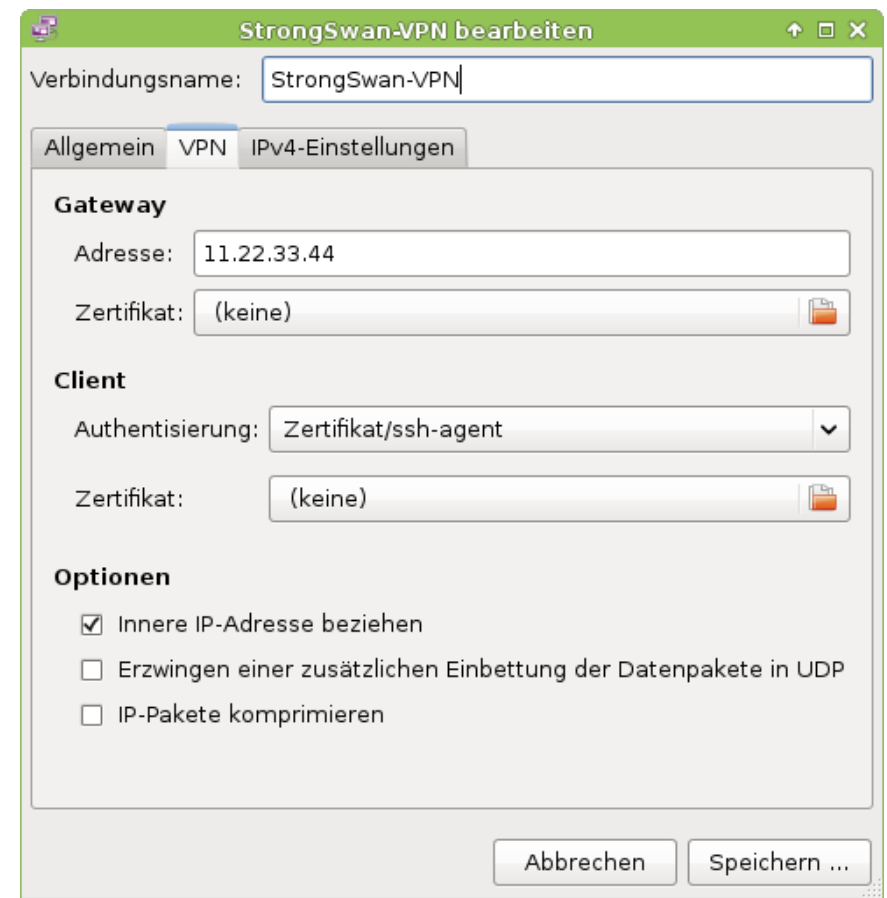


2.3 OpenPGP Smartcards mit Git verwenden

- ✓ Sichere Authentifizierung für Zugriffe mit SSH-Schlüssel
- ✓ Signieren von Commits mit OpenPGP-Signatur
 - (Linux Kernel Devs verwenden seit Kurzem Nitrokey zum Signieren.)
 - (Mozilla signiert Release Package mit Cryptostick, dem Nitrokey Vorläufer.)

2.4 StrongSwan-VPN Client Authentifizierung

- StrongSwan kann SSH Keys zur Authentifizierung von Clients nutzen.
- Der SSH public Key wird auf dem Server hinterlegt
- Client nutzt „ssh-agent“ für den Zugriff auf den private Key
- „ssh-agent“ wird von „gpg-agent“ bereitgestellt (s.o.)



2.5 Root-CA Zert. mit OpenPGP Smartcard

- Mit dem Tool XCA kann man ein kleine Certification Authority erstellen und X.509 Zertifikate verwalten
- Der private Signaturschlüssel der Root-CA ist dabei besonders zu schützen
- Man kann den privaten Schlüssel auf der OpenPGP Smartcard für den Signaturschlüssel nutzen
 - (1) Neue Datenbank für die CA anlegen
 - (2) OpenSC als PKCS#11 Provider laden
 - (3) Chipkarten Verwaltung öffnen und Signaturschlüssel importieren
 - (4) Selbstsigniertes Root-CA Zertifikat mit diesem Schlüssel erstellen
 - (5) X.509 Server- und Client-Zertifikate mit diesem Root-CA Zert. signieren

2.5 Root-CA Zert. mit OpenPGP Smartcard

X Certificate and Key management

XCA Optionen

Pflichtfelder im subject-name

countryName

Standard Hash Algorithmus

Zeichenkettentyp

Erstellungs- und Importnachrichten unterdrücken

Abgelaufene Zertifikate nicht farblich markieren

PKCS#11 anbieter

/usr/lib/i386-linux-gnu/opensc-pkcs11.so

X Certificate and Key management

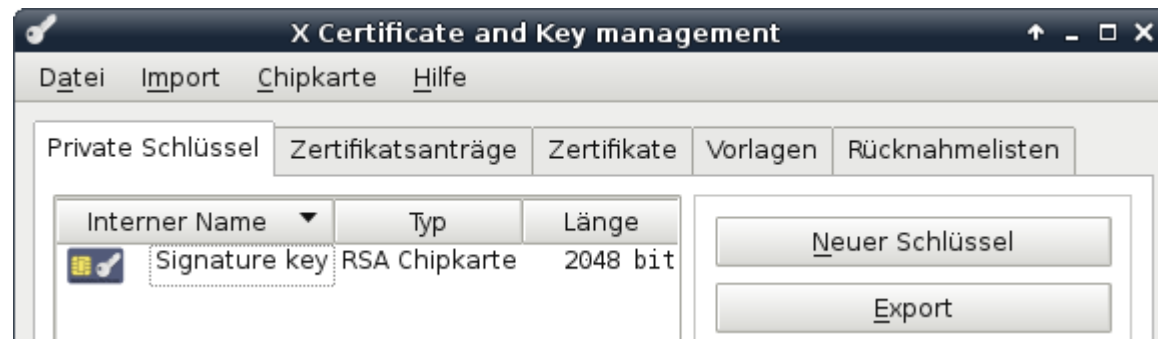
Chipkarte verwalten

Signature key

Cryptoki version: 2.20
Manufacturer: OpenSC (www.opensc-project.org)
Library: Smart card PKCS#11 API (0.0)

Name: OpenPGP card (User PIN (sig))
Modell: PKCS#15 emulated
Seriennummer: 000500000302

2.5 Root-CA Zert. mit OpenPGP Smartcard



2.6 OpenPGP Smartcards mit gpgsm nutzen

- X.509 Zertifikat erzeugen, dass den privaten Schlüssel auf der OpenPGP Smartcard als private Key nutzt:

Beispiel: ein Zertifikat zur Authentifizierung:

```
> gpgsm --gen-key > my-cert-request.csr
```

Bitte wählen Sie, welche Art von Schlüssel Sie möchten:

- (1) RSA
- (2) Vorhandener Schlüssel
- (3) Vorhandener Schlüssel auf der Karte

Ihre Auswahl? **3**

Vorhandene Schlüssel:

- (1) 001261F52353B286F49EC OPENPGP.1
- (2) 4ED102F55994645D57B00 OPENPGP.2
- (3) E16A99E2EDA7985634664 OPENPGP.3

Ihre Auswahl? **3**

2.6 OpenPGP Smartcards mit gpgsm nutzen

Mögliche Vorgänge eines RSA-Schlüssels:

- (1) signieren, verschlüsseln
- (2) signieren
- (3) verschlüsseln

Ihre Auswahl? **2**

Bitte geben sie den Namen des X.509 Subjekts ein:

```
CN="<Name>",OU="<Abteilung>",O="<Firma>",L="<Stadt>",ST="<Bundesland>",C=DE
```

...

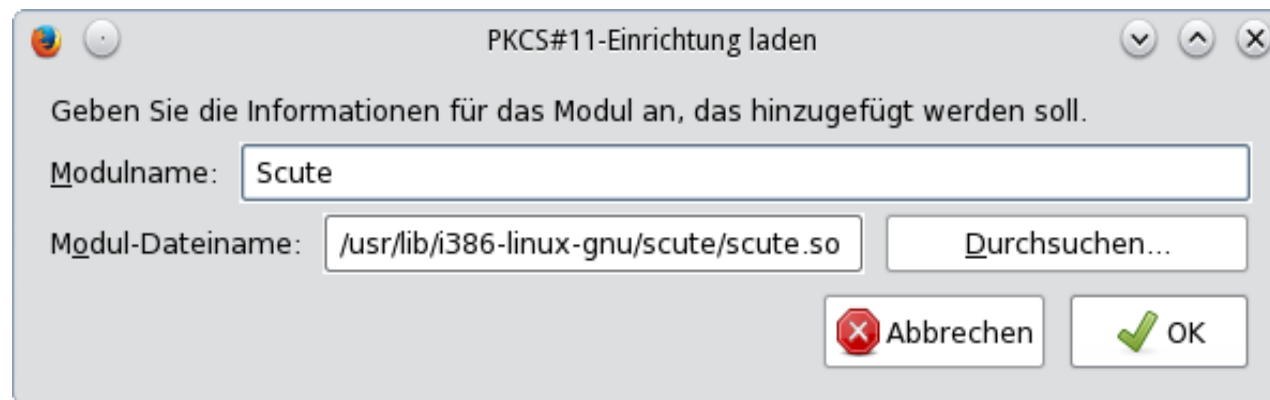
Fertig. Sie sollten nun den Request `my-cert-request.csr` an die CA senden.

2.6 OpenPGP Smartcards mit gpgsm nutzen

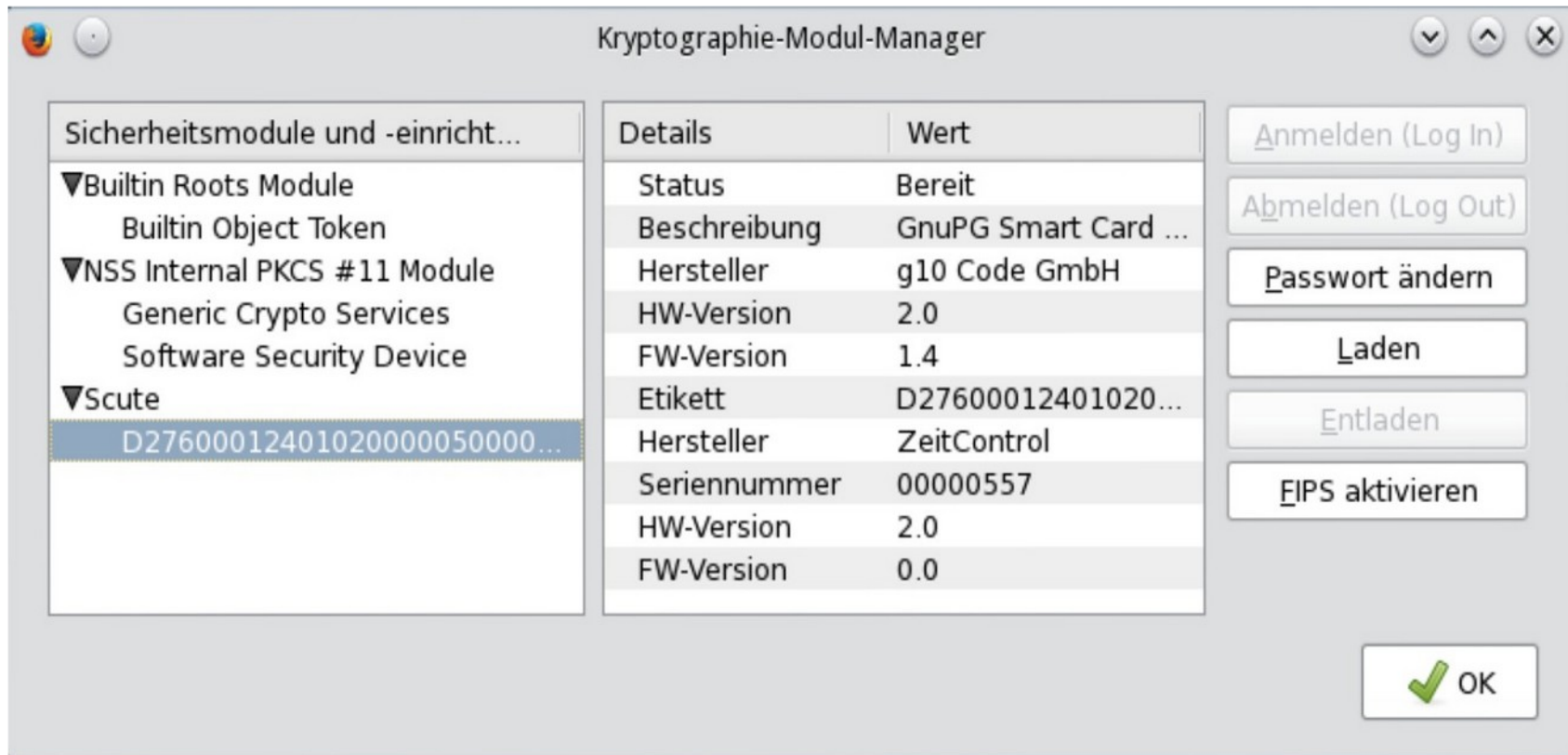
- Der CSR ist von einer Certification Authority zu signieren.
- Das von der CA signierte Zertifikat ist zu importieren:
 - > `gpgsm -import mein-cert.crt`

2.6 OpenPGP Smartcards mit gpgsm nutzen

- Das Programm, welches das Zertifikat für Authentifizierung nutzt, muss als PKCS#11 Provider das `scute` Modul laden
 - Firefox: „Einstellungen → Erweitert → Zertifikate → Krypto-Module → Laden“



2.6 OpenPGP Smartcards mit gpgsm nutzen



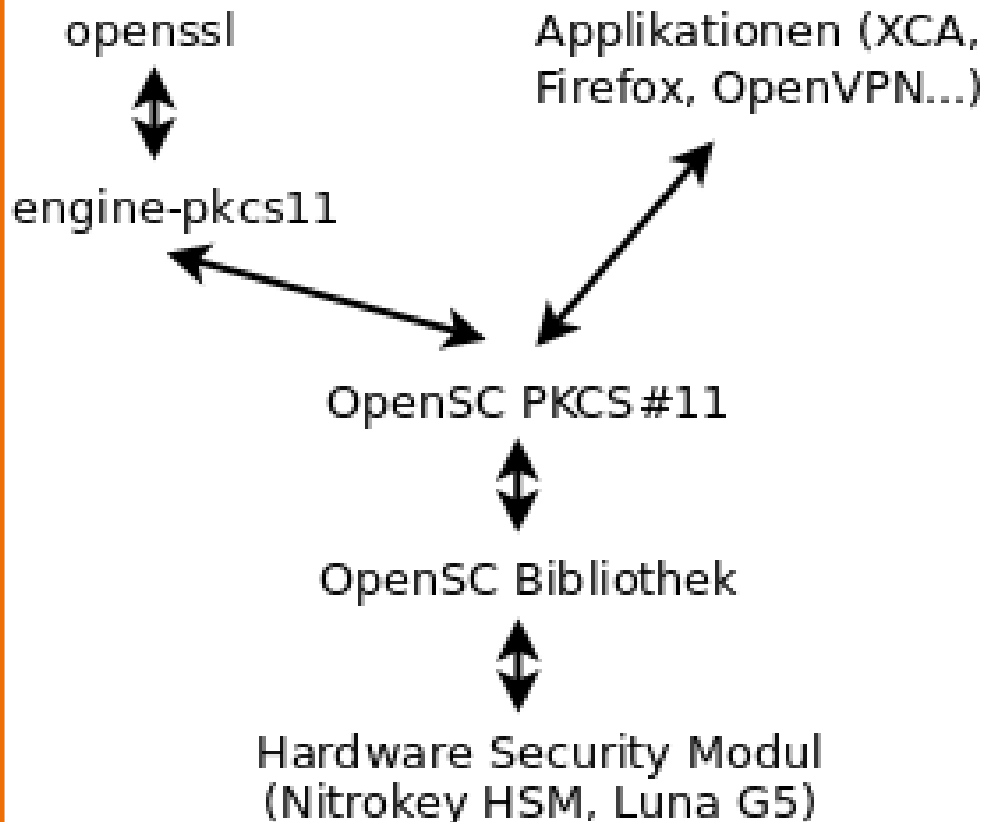
3. Hardware Security Module für X.509

- Die privaten Schlüssel sind auf dem HSM gespeichert
- Zusätzlich kann das Zertifikat auf dem HSM gespeichert werden
- M-aus-N Zugriff auf das HSM ist i.d.R. konfigurierbar (Vier-Augen-Prinzip)

3. Hardware Security Module für X.509

- Die privaten Schlüssel sind auf dem HSM gespeichert
- Zusätzlich kann ein Zertifikat auf dem HSM gespeichert werden
- M-aus-N Zugriff auf das HSM ist i.d.R. konfigurierbar (Vier-Augen-Prinzip)
- Anwendungen:
 - Verschlüsseln/Signieren von E-Mails (S/MIME)
 - Autorisieren von Usern, Zugriffsrechte verwalten...usw.
 - Als Root-Zertifikat für Firmen PKI/CA (z.B. mit Vier-Augen-Prinzip)
 - Device Authentication (embedded) mit Secure Message Channel
- Produkte:
 - Nitrokey HSM
 - Gemalto Luna G5 (USB) oder Luna SA (Netzwerk Interface)

3. Hardware Security Module für X.509



- OpenSC stellt einen PKCS#11 Provider bereit, der den Zugriff auf das HSM anbietet
- Jede Anwendung, die diese Schlüssel/Zertifikate nutzen will, muss diesen PKCS#11 Provider importieren. (Firefox, Thunderbird, XCA, openssl...)
- Verwaltung des HSM erfolgt z.B. mit der pkcs15-Toolbox (Kommandozeilentools)

3.1 X.509 HSMs vorbereiten mit pkcs15-init

→ Initialisierung eines X.509 HSM:

```
> pkcs15-init --erase-card
> pkcs15-init --create-pkcs15 [--so-pin 0000 --so-puk 111111111111]
> pkcs15-init --store-pin --id 01 --label „Mustermann“
    [--pin 0000 --puk 111111111111]
```

→ Schlüssel auf dem X.509 HSM generieren:

```
> pkcs15-init --generate-key rsa/2048 --auth-id 01
```

→ Vorhandene Keys und Zertifikate importieren:

```
> pkcs15-init --auth-id 01 --store-private-key myKey.pem
> pkcs15-init --auth-id 01 --store-certificate myCert.pem
```

```
> pkcs15-tool --list-pins --list-keys --list-certificates
```

3.2 X.509 HSMs mit openssl nutzen

(1) PKCS#11 Provider für OpenSC dynamisch laden:

```
> openssl
OpenSSL> engine -t dynamic -pre SO_PATH:/usr/lib/engines/libpkcs11.so \
                -pre ID:pkcs11 -pre LIST_ADD:1 -pre LOAD \
                -pre MODULE_PATH:/usr/lib/i386-linux-gnu/opensc-pkcs11.so
```

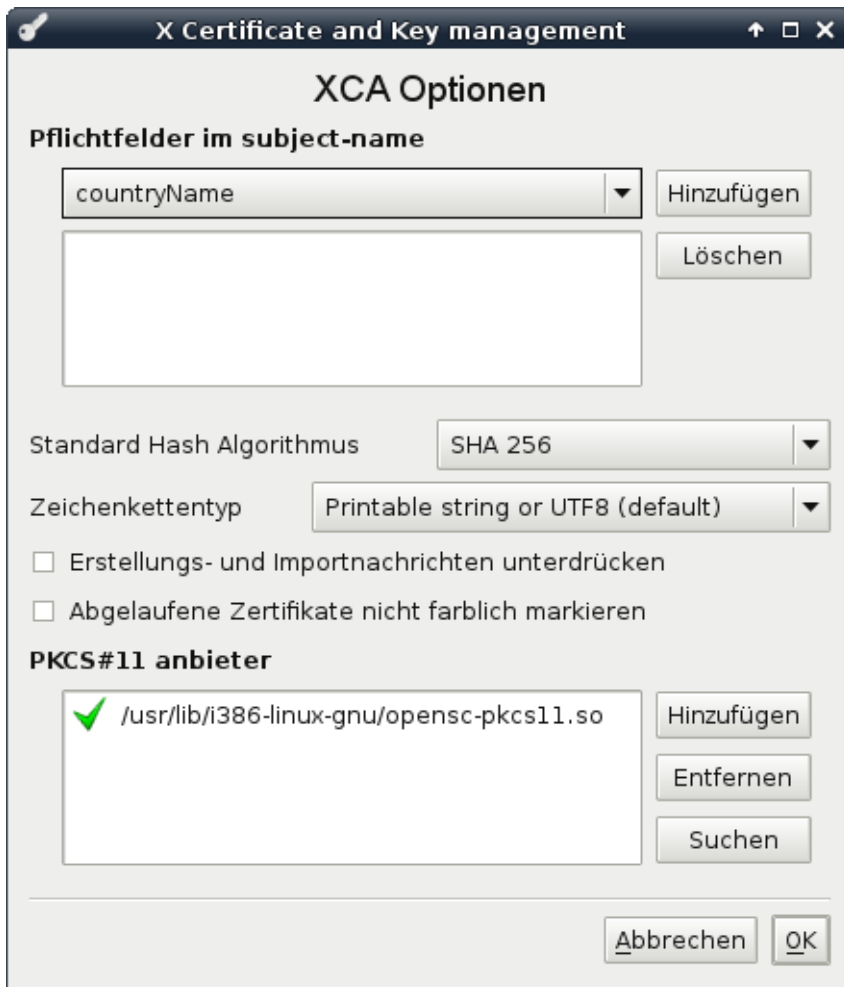
(2) Einen CSR mit dem privaten Schlüssel des HSM erstellen:

```
OpenSSL> req -new -out req.pem -text -engine pkcs11 -keyform engine \
            -key "pkcs11:object=ca-key;type=private;pin-value=XXXX" \
            -subj "/CN=.../OU=.../O=.../L=.../ST=.../C=DE" -x509
```

(3) Certificate Request signieren (z.B. selbstsigniertes CA-Zert):

```
OpenSSL> x509 -in req.pem -out cert.pem -engine pkcs11 -keyform engine \
            -signkey "pkcs11:object=ca-key;type=private;pin-value=XXXX"
```

3.2 X.509 HSMs mit openssl und XCA nutzen



X Certificate and Key management

XCA Optionen

Pflichtfelder im subject-name

countryName

Standard Hash Algorithmus

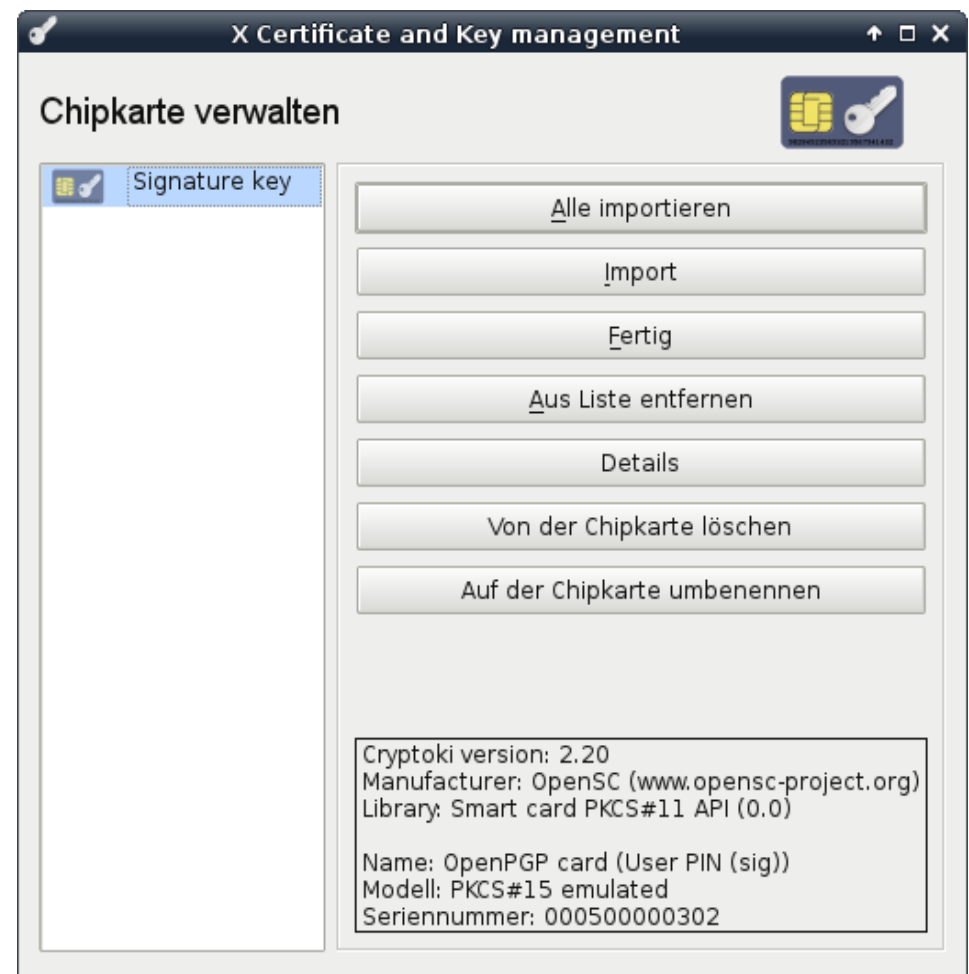
Zeichenkettentyp

Erstellungs- und Importnachrichten unterdrücken

Abgelaufene Zertifikate nicht farblich markieren

PKCS#11 anbieter

/usr/lib/i386-linux-gnu/opensc-pkcs11.so



X Certificate and Key management

Chipkarte verwalten

Signature key

Cryptoki version: 2.20
Manufacturer: OpenSC (www.opensc-project.org)
Library: Smart card PKCS#11 API (0.0)

Name: OpenPGP card (User PIN (sig))
Modell: PKCS#15 emulated
Seriennummer: 000500000302

3.3 X.509 HSMs mit Firefox und Thunderbird

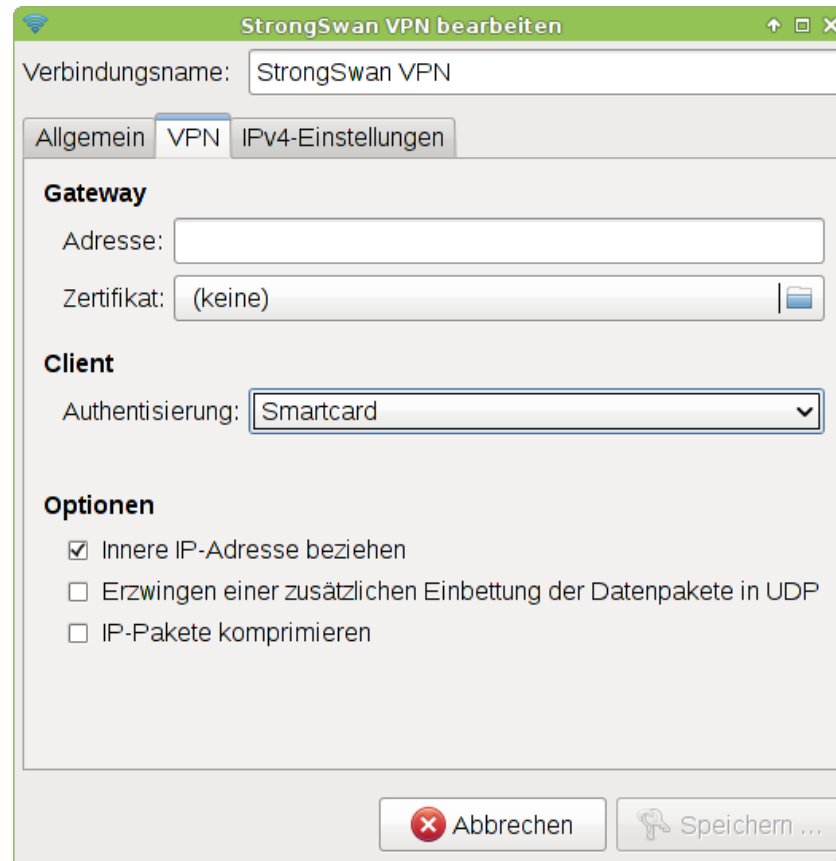
OpenSC PKCS#11 Provider als Kryptografie Modul laden

- Windows: C:\Programme\Smart card bundle\opensc-pkcs11.dll
- Linux: /usr/lib/i386-linux-gnu/opensc-pkcs11.so



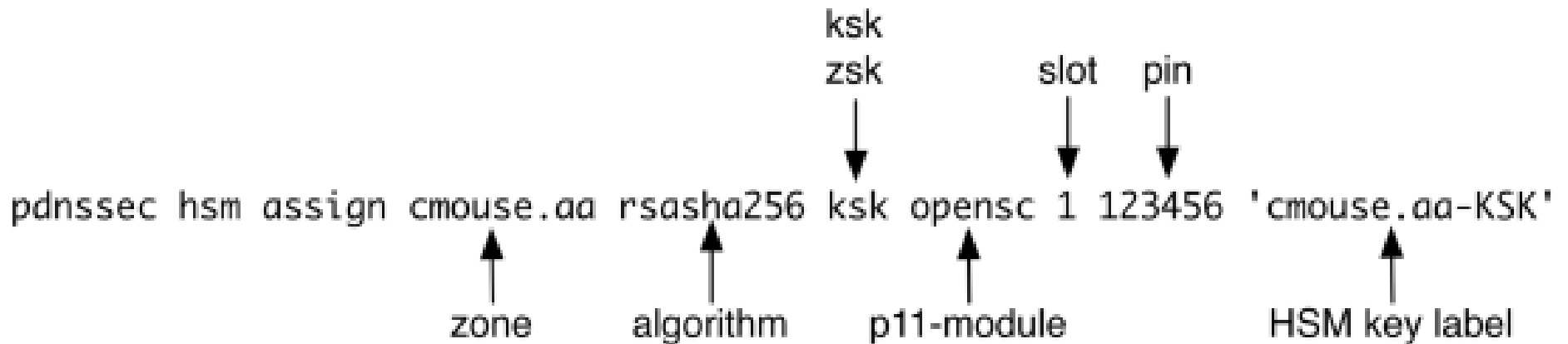
The screenshot shows the Firefox settings interface. The left sidebar has the 'Erweitert' (Advanced) option selected and circled in yellow. The main content area is titled 'Erweitert' and has several sub-tabs: 'Allgemein', 'Datenübermittlung', 'Netzwerk', 'Update', and 'Zertifikate'. The 'Zertifikate' tab is selected and circled in yellow. Below the tabs, there are options for 'Anfragen' (Requests) when a website asks for a personal security certificate: 'Automatisch einwählen' (selected) and 'Jedes Mal fragen' (selected). There is also a checked option for 'Aktuelle Gültigkeit von Zertifikaten durch Anfrage bei OCSP-Server bestätigen lassen'. At the bottom, there are two buttons: 'Zertifikate anzeigen' and 'Kryptographie-Module', with the latter circled in yellow. An orange arrow points from the 'Erweitert' sidebar item to the 'Zertifikate' tab, and another orange arrow points from the 'Kryptographie-Module' button to the 'Zertifikate' tab.

3.4 X.509 HSM mit StrongSwan für Client Auth



3.5 X.509 HSM für DNSSEC Signaturschlüssel mit PowerDNS verwenden

```
pdnssec hsm assign cmouse.aa rsasha256 ksk opensc 1 123456 'cmouse.aa-KSK'
```



zone algorithm p11-module HSM key label

ksk zsk slot pin

Vielen Dank für die Aufmerksamkeit

- Für Interessierte habe ich paar Nitrokeys zum Verkauf:
 - Nitrokey Pro: OpenPGP Smartcard + OTP und Passwortspeicher 49,-€
 - Nitrokey HSM: für 48 RSA Keys (2048 Bit) und 60 ECC Keys: 49,- €