eToken unter Linux

Computerlabor@KuZeB
13. Dezember 2010

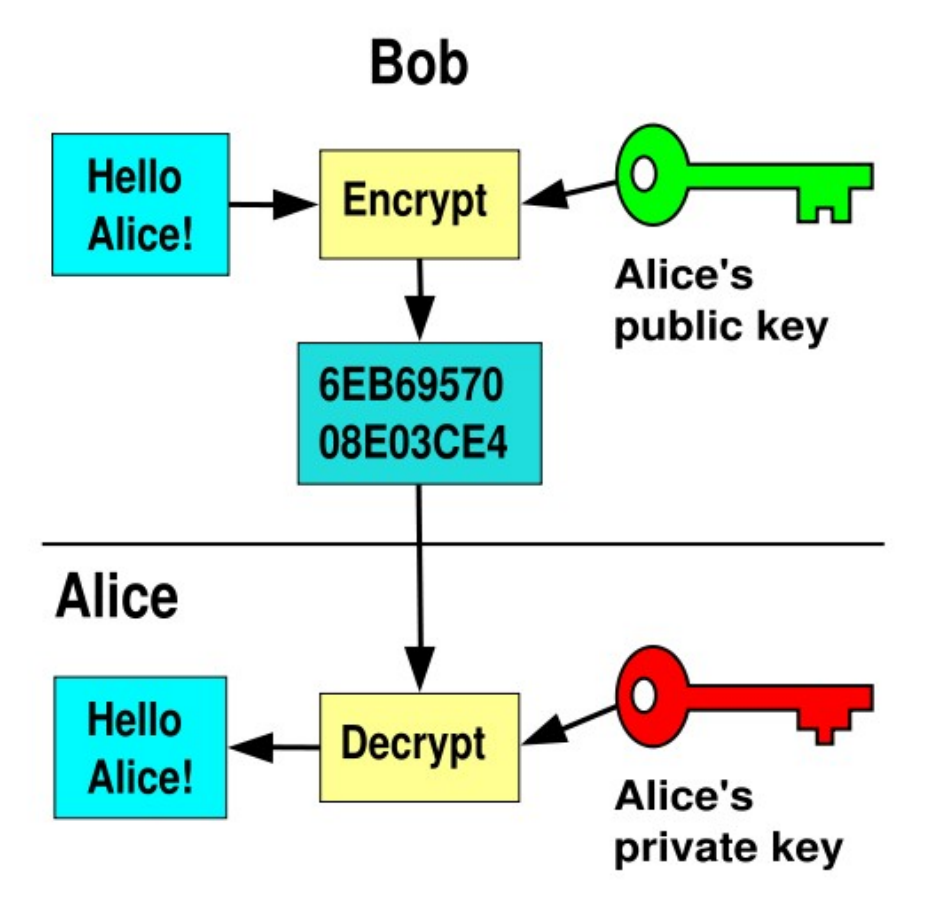
Aufbau

- Zuerst ein bisschen Theorie
- Warum überhaupt so ein eToken?
- Praktischer Teil unter Linux

Grundsätzliches

- Das eToken hat einen eigenen Prozessor
- Der Schlüssel wird direkt auf dem Token generiert
- Der Schlüssel kann das Token NICHT mehr verlassen
- Das eToken ist eine Smartcard+”Kartenleser” in Form eines USB-Sticks

Überblick Public-Key Verschl.



- Aladdin eToken unterstützen RSA
- Benutzt Primzahlen, Hochrechnung und modulo-Operator
- Keine Mathestunde ;)

Motivation

- Authentifizierung
- Verschlüsselung

Authentifizierung

- Passwort
- Keyfile
- Etoken

Passwort

- Single-factor
- Kann weitergesagt werden, Shouldersurfing, Hardware-Keylogger etc

Keyfile

- Two-factor
- Eine Datei plus ein Passwort
- Nachteil: Keyfile kann von Attacker (oder malicious user!) kopiert werden, Passwort lässt sich wie vorhin gesagt kriegen

eToken

- Two-factor
- Ein USB-Stick plus Passwort
- Vorteil: Attacker müsste physikalisch im Besitz des Sticks sein. Lässt sich nicht kopieren, somit ist sichergestellt dass nur exakt eine Version im Umlauf ist
- Somit lassen sich Benutzerrechte durch Einziehen des Stick entziehen

Motivation

- Authentifizierung
- Verschlüsselung

Public-Key Verschlüsselung

- Vorteile ähnlich wie bei Authentifizierung
- Kopieren des privaten Schlüssels wird verhindert
- Benutzungsrecht eines Schlüssels lässt sich durch Einziehen des Sticks entziehen (Vereins-Mailadresse etc)

Smartcard-Standards

- RSA hat den PKCS#11/PKCS#15 Standard entwickelt
  - PKCS#11: Schnittstelle für die Smartcard
  - PKCS#15: Dateisystem auf der Smartcard

Smartcards unter Linux

- OpenSC ist ein offenes Smartcard Framework unter Linux
  - Bindet Smartcards per PKCS#11 Schnittstelle (oder per OpenCT) ein, verwaltet die Daten auf der Smartcard mit dem PKCS#15 Filesystem
- OpenCT ist ein offenes Cardreader Framework
  - Leider kein offener Treiber für eTokens, daher benutzen wir den Aladdin PKCS#11 Treiber

Übersicht Einrichten

- Benötigte Software installieren
- Stick per PKCS#11 mit dem Aladdin-Treiber initialisieren
- PKCS#15 Filesystem erstellen
- User und Key erstellen, Key signieren
- Profit!

Wichtig

- eToken jetzt vom Laptop abziehen, sonst wird es nach Installation der Software nicht automatisch erkannt

Terminologie

- SO/Security Officer: ”Admin” des eTokens. Darf Datenstruktur des eTokens verändern etc
- User PIN: Passwort des Benutzers. Wichtig: obwohl es PIN heisst können auch Buchstaben verwendet werden!
- PUK: Personal Unblocking Key. Zum wieder freischalten des Sticks nach zuviel falsch eingebenen Passwörtern
- SO und User PIN können gleich sein

Benötigte Software

- ”apt-get install pcscd opensc openct openssl libengine-pkcs11-openssl libqt4-core libqt4-gui”
- Bei Fehlermeldungen bitte den unteren Befehl noch **nicht eingeben**!
- ”dpkg -i pkiclient*”
- Konfiguration der maximalen Passwortversuche:
  - ”nano /usr/share/opensc/pkcs15.profile ”

eToken initialisieren

- ”pkcs11-tool --module /usr/lib/libeTPkcs11.so --init-token --label dominik”

PKCS#15 Filesystem erstellen

- ”pkcs15-init --create-pkcs15”
- PUK/unblocking key: Wird benötigt wenn das Passwort zu viel mal falsch eingegeben wurde. Kann weggelassen werden (und hab ich bei mir persönlich auch so gemacht, dafür 10 Passwort retries)

User und Key erstellen

- ”pkcs15-init --store-pin --auth-id 01 --label "Dominik Russenberger" ”
- ”pkcs15-init --generate-key rsa/2048 --auth-id 01”

Zertifikat signieren

- Etoken normalerweise für Einsatz in grösseren Unternehmen gedacht
- Zertifizierungsstelle wie bei HTTPS-Verbindungen
- Wir machen kurzerhand unser eigenes Zertifikat

Signieren, praktischer Teil

- ”openssl”
- Jetzt befinden wir uns in der SSL-Shell
  - ”engine dynamic -pre SO_PATH:/usr/lib/engines/engine_pkcs11.so -pre ID:pkcs11 -pre LIST_ADD:1 -pre LOAD -pre MODULE_PATH:opensc-pkcs11.so”
  - ”req -engine pkcs11 -new -key id_45 -keyform engine -x509 -out cert.pem -text ”
- Ctrl+D zum beenden

Zertifikat hochladen

- ”pkcs15-init --store-certificate cert.pem --auth-id 01 --id 45 --format pem ”