Yubikey als One Time Password für PAM by Peter Murr

Über Yubikeys

Die Yubikeys der Firma Yubico [http://www.yubico.com] dienen zur robusten Authentifizierung von Netzwerken und Diensten für verschiedene Plattformen und benötigt keinerlei Client Software (Treiber). Das revolutionäre Gerät wird in den USB-Port gesteckt. Eine kurze Berührung des Yubikey reicht aus, um jedesmal die Identität des Benutzers und ein einmaliges Passwort (One Time Password = OTP) zu senden.

Der unvergleichliche Preis (~31,25$ inkl. Steuer), die Benutzerfreundlichkeit und das Open-Source-Geschäftsmodell führen dazu, dass der Yubikey derzeit sehr rasch von Unternehmen, Software-Anwendungen und Internet-Diensten angenommen wird.

Hinweise

Die Yubikeys der neueren Generation (V2 und höher) können mit dem sog. Personalization Tool [http://yubico.com/developers/personalization/] reprogrammiert werden. Desweiteren unterstützen sie nun auch neben dem OTP (One Time Password), welches durch kurzes Drücken auf den UbiKey ausgegeben wird auch die Option ein statisches Passwort zu hinterlegen. Dieses wird ausgegeben, wenn der UbiKey länger als 1.5 Sekunden gedrückt wird. Die neueren UbiKeys haben also 2 unabhängige Speicherorte eingebaut, die frei programmiert (OTP oder statisch) werden können.

Durch Reprogrammierung kann auch Open Authentication (OATH), ein offener Standard der von VeriSign entworfen wurde, um robuste Authentifizierung für Geräte unterschiedlicher Hersteller zu ermöglichen, realisiert werden.

Es gibt auch Yubikeys, die zusätzlich zu den 2 OTP-Bereichen, einen RFID-Chip eingebaut haben, somit kann man den Yubikey als Authentifizierungsgerät für alle möglichen Dienste verwenden.

Yubikeys können mit einem benutzerspezifischen Firmenlogo bestellt werden. Auch die Papiereinlage der Verpackung (enthält standardmässig den Aufdruck Yubico), kann bereits ab Werk vordefiniert werden.

Die Passwortüberprüfung der Ubikeys erfolgt prinzipiell immer über entsprechende Online-Dienste. Dazu hat die Firma Yubico einen Server installiert, gegen den man die Schlüssel prüfen kann. Dieser Dienst ist sehr redundant aufgebaut. Details dazu können hier [http://www.yubico.com/files/Validation_Service_Reliability_2009-09-09.pdf] nachgelesen werden.

Für verschiedene Betriebssysteme und Platformen werden seitens Yubico und der Yubico-Developer-Community verschiedenste Möglichkeiten der Implementierung angeboten. Hier eine kurze Liste mit wichtigen Features.

- Web Service Clients
- Web Service API
- Basic Server Library
- Validation Server
- OpenID Server
- SAML Server

Um einen eigenen "Validation Server" zu betreiben, muss man die AES Schlüssel der UbiKeys importieren. Wie das geht, kann man hier [http://yubico.com/developers/srv/] nachlesen.

Diese Dokumentation beschäftigt sich damit, wie man das One-Time-Password-Feature des UbiKeys auch ohne ständig notwendigen Kontakt zum Yubikey Validierungs-Server nutzen kann, um Benutzer zu authentifizieren.

Als Beispiel soll eine Authentifizierung für einen SSH-Login per UbiKey OTP realisiert werden. Vermutlich lässt sich mit diesem HowTo auch ein Radius Server realisieren, somit stehen alle Möglichkeiten offen. Es gibt auch diverse HowTos im Internet.

Installation des Debian Test-Systems (Lenny)

Für diese Dokumentation wurde ein Debian System (Lenny) in einer virtuellen Maschine aufgesetzt, deshalb ist diese Anleitung auch speziell an eine Debian-Linux Distribution angepasst, andere Distributionen sollten jedoch in ähnlicher Art und Weise konfigurierbar sein.

Installation des Betriebssystems

Das Sytem wurde von der **Debian Netzinstallations Minimal-CD installiert.** und während des Setups direkt per Proxy und Internet in englischer Sprache und deutschem Tastaturlayout installiert. Es wurde nur die Option "Standard System" gewählt, d.h. ein System ohne X-Server. Nach der Installation wurde das Netzwerk konfiguriert und das System auf aktuellen Stand gebracht. Der Hostname für diese Dokumentation lautete **lenny-otp**

Installation zusätzlicher Komponenten

```
aptitude install autoconf automake binutils ccp gcc linux-headers-$(uname -r) make psmisc
aptitude install ssh checkinstall libusb-dev libpam-dev
```

Ggf. (falls auf dem OS möglich) sollte auch libusb in Version 1.0 installiert werden. Dies war bei Debian Lenny aus den Standard Repositories nicht möglich.

```
aptitude install libusb-1.0-0 libusb-1.0-0-dev
```

Weitere zusätzliche Installationen sind in den entsprechenden Unterpunkten dieser Anleitung dokumentiert.

Kompilieren von YubiPAM

Vorbereitung

Zunächst installieren wir uns libpam-dev (Development files for PAM), falls nicht bereits vorhanden:

```
aptitude install libpam-dev
```

YubiPAM herunterladen - kompilieren - Paket erstellen

Zunächst laden wir uns den Quellcode von **YubiPAM** herunter

```
  mkdir /tmp/yubikey
  cd /tmp/yubikey
  wget http://www.securixlive.com/download/yubipam/YubiPAM-1.0.4.tar.gz
  tar xzvf YubiPAM-1.0.4.tar.gz
  cd YubiPAM-*
  ./configure
  checkinstall --maintainer=pcfreak@pcfreak.de --install=no -D make install
  checkinstall --pkgversion=1.0.4-lenny --maintainer=pcfreak@pcfreak.de --install=no -D make install
```

Als Ergebnis sollten wir

```
#*********************************************************************#
 Done. The new package has been saved to

 /yourpath/yubipam_1.0.4-1_i386.deb
 You can install it in your system anytime using:

      dpkg -i yubipam_1.0.4-1_i386.deb
#*********************************************************************#
```

erhalten.

Hinweis
Wer anstatt des Default Prompts

```
Yubikey OTP:
```

bei einer Anmeldung (z.B. per ssh) einen anderen Prompt haben will, z.B.

```
Please press your UbiKey:
```

der kann im Unterordner /src die Datei

```
pam_yubikey.c
```

vor dem Kompilieren anpassen. Die entsprechende Stelle in pam_yubikey.c ist:

```
          /* prompt for the Yubikey OTP */
          {
                  otp = get_response(pamh, "Yubikey OTP: ", verbose_otp);
                  retval = pam_set_item(pamh, PAM_AUTHTOK, otp);
          }
```

Installation von YubiPAM

Zur Installation von YubiPAM benötigen wir das vorher erstellte Debian Paket (yubipam_1.04-1_i386.deb).[1)]

```
dpkg -i yubipam_1.0.4-1_i386.deb
Selecting previously deselected package yubipam.
(Reading database ... 30939 files and directories currently installed.)
Unpacking yubipam (from yubipam_1.0.4-1_i386.deb) ...
Setting up yubipam (1.0.4-1) ...
```

YubiPAM vorkonfigurieren

YubiPAM liefert eine ordentliche Dokumentation. Diese findet man nach der Installation unter

```
/usr/share/doc/yubipam
```

Wir führen zunächst die wichtigen Schritte gemäß der Datei **INSTALL** durch:

Gruppe yubiauth erstellen

```
addgroup yubiauth
Adding group `yubiauth' (GID 1001) ...
Done.
```

Leere Datenbank erstellen

```
touch /etc/yubikey
```

Ändern der Gruppe für "/etc/yubikey" und "/sbin/yk_chkpwd" um der Gruppe yubiauth das gemeinsame Editieren der Datenbank zu erlauben

```
chgrp yubiauth /etc/yubikey /sbin/yk_chkpwd
```

Berechtigungen für /etc/yubikey setzen

```
chmod g+rw /etc/yubikey
```

Berechtigungen für /sbin/yk_chkpwd setzen

```
chmod g+s /sbin/yk_chkpwd
```

AES-Key Bereitstellung

Um eine Offline-Authentifizierung (kein Kontakt zum Yubico Internet Server) durchführen zu können, brauchen wir den AES-Key unseres Yubikeys. Dieser ist aber bei Auslieferungszustand nur der Firma Yubico bekannt. (in etwa vergleichbar mit public/private Key, wobei Yubico den Private Key hat)
Damit wir unseren AES-Key also ermitteln können, müssen wir ihn **neu setzen**, dann ist er uns logischerweise bekannt.

Die Reprogrammierung erfolgt mit dem sog. Personalization Tool von Ubico. Dieses gibt es für verschiedene Platformen, auch für Linux.

Kompilieren von libyubikey

Um den Yubikey unter Linux reprogrammieren zu können, benotigen wir das Tool **ykpersonalize**. Wir müssen es selbst kompilieren.
Voraussetzung für **ykpersonalize** ist **libyubikey**, die wir ebenfalls selbst kompilieren müssen, also los:

```
mkdir /tmp/libyubikey
cd /tmp/libyubikey
wget http://yubico-c.googlecode.com/files/libyubikey-1.5.tar.gz
tar xzvf libyubikey-1.5.tar.gz
cd libyubikey-*
./configure --libdir=/usr/lib
checkinstall --maintainer=pcfreak@pcfreak.de --install=no -D make install
```

Als Ergebnis sollten wir

```
#********************************************************************#
 Done. The new package has been saved to

 /yourpath/libyubikey_1.5-1_i386.deb
 You can install it in your system anytime using:

      dpkg -i libyubikey_1.5-1_i386.deb
#********************************************************************#
```

erhalten.

Installation von libyubikey

```
dpkg -i libyubikey_1.5-1_i386.deb
Selecting previously deselected package libyubikey.
(Reading database ... 30952 files and directories currently installed.)
Unpacking libyubikey (from libyubikey_1.5-1_i386.deb) ...
Setting up libyubikey (1.5-1) ...
```

Kompilieren von ykpers

```
mkdir /tmp/ykpers
cd /tmp/ykpers
wget http://yubikey-personalization.googlecode.com/files/ykpers-1.1.tar.gz
tar xzvf ykpers-1.1.tar.gz
cd ykpers-*
```

Falls auf unserem System libusb-1.0 verfügbar ist, konfigurieren wir mit

```
./configure --with-backend=libusb-1.0
```

ansonsten mit

```
./configure
```

Dann gehts weiter mit der Paketerstellung

```
checkinstall --maintainer=pcfreak@pcfreak.de --install=no -D make install
```

Als Ergebnis erhalten wir

```
#********************************************************************#
 Done. The new package has been saved to

 /yourpath/ykpers_1.1-1_i386.deb
 You can install it in your system anytime using:

      dpkg -i ykpers_1.1-1_i386.deb
#********************************************************************#
```

Installation von ykpers

```
dpkg -i ykpers_1.1-1_i386.deb
Selecting previously deselected package ykpers.
(Reading database ... 30968 files and directories currently installed.)
Unpacking ykpers (from ykpers_1.1-1_i386.deb) ...
Setting up ykpers (1.1-1) ...
Processing triggers for man-db ...
```

AES Key reprogrammieren [Linux]

Erstellen eines Public Keys

Hinweis

Es kann zu Problemen mit dem Personalization Tool unter Linux kommen. Die Fehlermeldung lautet dann ungefähr so:

```
USB error: could not claim interface 0: Device or resource busy
```

Wie man in einschlägigen Foren nachlesen kann, hat das etwas mit dem **usbhid** Layer zu tun. Mehr dazu hier [http://code.google.com/p/yubikey-core/source/browse/trunk/docs/quirks-and-workarounds.txt]
Der dort gelistete Workarrount 2 funktioniert recht gut, jedoch kann man wie ebenso beschrieben auch mit einem entsprechenden Verweis auf libusb-1.0-0 kompilieren um das Problem zu umgehen. Für den Workarround 2 habe ich auch ein kleines Script geschrieben.
Man steckt den Yubikey an den Rechner an und führt dann folgenden Befehl aus:
Ubuntu 9.10

```
  UBI_ID=$(grep -E ".*Yubico Yubikey.*[0-9]-[0-9]:[0-9]\.[0-9].*" /var/log/messages | sed -e 's/^.*\([0-9]-[0-9]:[0-9]\.[0-9]\).*/\1/g'| tail -n 1)
```

Die Eingabe von

```
  echo $UBI_ID
```

sollte einen String in folgendem Format ausgeben

```
1-1:1.0
```

Nun haben wir die aktuelle Device-ID des Yubikeys in der Variable **UBI_ID**. Nun lösen wir die Bindung von usbhid für dieses Gerät: Das ist nur ein Quick&Dirty Hack und ist sicherlich optimierbar - funktionierte aber bei mir.

```
  echo $UBI_ID > /sys/bus/usb/drivers/usbhid/unbind
```

Debian Lenny
Für Debian Lenny in einer virtuellen Maschine (in die alle Input Devices durchgeschalten wurden) funktioniert leider nur Workarround 1

```
rmmod usbhid && modprobe usbhid quirks=0x1050:0x0010:0x04
```

Nun sollte der Yubikey auch unter Linux mit **ykpersonalize** ansprechbar sein und wir können fortfahren.

Zur Erstellung eines Private Keys habe ich ein kleines Script geschrieben, welches durch Benutzung von **modhex** (im ykpers Paket)[2)] einen entsprechend weiterverarbeitbaren Key erzeugt.

```
  #!/bin/bash
  #yubiprogramprepare for Yubikey by pcfreak [at] pcfreak [dot] de
  #needed for my documentation on re-programming Yubikeys for YubiPAM
  FILENAME=$PWD/FILE$RANDOM
  LOGFILE=$PWD/yubiprogramprepare.log
  echo -e "Please enter 6 characters for uid:"
  read -n6 INPUT1
  echo -e ""
  echo -e "uid: $INPUT1" >> $LOGFILE

  echo -e "Please enter 5 characters for public name:"
  read -n5 INPUT2
  echo -e ""
  echo -e "pub: $INPUT2" >> $LOGFILE
  echo -e "out: $FILENAME" >> $LOGFILE

  MODHEX1=$(modhex -h ff)$(modhex $INPUT2)

  echo -e "public name ModHex encoded with ff prefix:\t"$MODHEX1
  echo -e "public name    Hex encoded with ff prefix:\t"$(modhex -d -h $MODHEX1)

  HEX2=$(echo -e "$INPUT2"|hexdump -v -e '1/1 "%02x"')

  echo -e "uid            Hex encoded               :\t"$HEX2
  echo -e ""
  echo -e "run1:  echo -e "\f"|sudo ykpersonalize -s$FILENAME -ouid=$HEX2 -ofixed=$MODHEX1"
  echo -e "run1:  echo -f "\f"|sudo ykpersonalize -s$FILENAME -ouid=$HEX2 -ofixed=$MODHEX1" >> $LOGFILE
  echo -e ""
  echo -e "run2:  sudo ykpersonalize -ouid=$HEX2 -ofixed=$MODHEX1 -a\$(grep -E \"^key: h:.*$\" $FILENAME|sed -e 's/^.*:.*:\(.*\)/\1/g')"
  echo -e "run2:  sudo ykpersonalize -ouid=$HEX2 -ofixed=$MODHEX1 -a\$(grep -E \"^key: h:.*$\" $FILENAME|sed -e 's/^.*:.*:\(.*\)/\1/g')" >> $LOGFILE
  echo -e ""
  echo -e ""
  echo -e "You should keep a copy of $FILENAME and $LOGFILE for later use!"
```

Wir speichern obiges Skript als "yubiprogramprepare" irgendow im Filesystem ab und machen es mit

```
chmod +x yubiprogramprepare
```

ausführbar. Nun rufen wir yubiprogramprepare auf. In diesem Beispiel wurden die Eingaben "123456" und "12345" gemacht. Diese sind beispielhaft und sollten durch die gewünschten Eingaben ersetzt werden.

Das Ergebnis eines Aufrufs von ./yubiprogramprepare sollte etwa wie folgt aussehen:

```
./yubiprogramprepare
Please enter 6 characters for uid:
> 123456
Please enter 5 characters for public name:
> 12345
public name ModHex encoded with ff prefix:      vvebedeeefeg
public name    Hex encoded with ff prefix:      ff3132333435
uid            Hex encoded               :      31323334350a

run1:  echo -e f|sudo ykpersonalize -s/root/yubikey/yubiprogramprepare/FILE23535 -ouid=31323334350a -ofixed=vvebedeeefeg

run2:  sudo ykpersonalize -ouid=31323334350a -ofixed=vvebedeeefeg -a$(grep -E "^key: h:.*$" /root/yubikey/yubiprogramprepare/FILE23535|sed -e 's/^.*:.*:\(.*\)/\1/

You should keep a copy of /root/yubikey/yubiprogramprepare/FILE23535 and /root/yubikey/yubiprogramprepare/yubiprogramprepare.log for later use!
```

Nun führen wir den bei run1 genannten Befehl aus (der sudo Befehl muss je nach Distribution weggelassen werden):

```
  echo -e f|sudo ykpersonalize -s/root/yubikey/yubiprogramprepare/FILE23535 -ouid=31323334350a -ofixed=vvebedeeefeg
```

und erhalten die Ausgabe

```
Firmware version 2.0.2 Touch level 1792 Program sequence 3
Passphrase to create AES key:
```

Der vorgeschaltete Echo Befehl bewirkt, dass die Abfrage nach dem Passwort für den AES-Key automatisch übersprungen wird und somit ein zufälliger Key generiert wird.
Anschliessend führen wir den bei run2 genannten Befehl aus (der sudo Befehl muss je nach Distribution weggelassen werden):

```
  sudo ykpersonalize -ouid=31323334350a -ofixed=vvebedeeefeg -a$(grep -E "^key: h:.*$" /root/yubikey/yubiprogramprepare/FILE23535|sed -e 's/^.*:.*:\(.*\)/\1/g')
```

und erhalten:

```
Firmware version 2.0.2 Touch level 1792 Program sequence 3
Configuration data to be written to key configuration 1:

fixed: m:vvebedeeefeg
uid: h:31323334350a
key: h:05311f79fc8d7b6cdf10abf86920203b
acc_code: h:000000000000
ticket_flags: APPEND_CR
config_flags:

Commit? (y/n) [n]: y
```

Wir bestätigen bei **Commit?** mit **y** Der Schlüssel wurde nun programmiert.
Nun unbedingt eine Kopie von

```
/root/yubikey/yubiprogramprepare/FILE23535  (Name von oben abschreiben)
```

und

```
/root/yubikey/yubiprogramprepare/yubiprogramprepare.log  (Name von oben abschreiben)
```

erstellen, da darin die zur Programmierung benutzten Werte enthalten sind.
Wir benötigen die dort enthaltenen Daten für das Hochladen des Keys zu Yubico, um die Online-Funktionalität wiederherzustellen.

Programmierung überprüfen

Mit dem Befehl **ykdebug** kan man die Programmierung des Yubikeys überprüfen, indem man ihn folgendermaßen aufruft.

```
ykdebug <AESKEY> <TOKEN>
```

Den Wert für <AESKEY> können wir aus unserem Konfigurationsfile (**anpassen - Name ist zufällig erzeugt**) auslesen. Er steht dort bei **key: h:**

```
cat FILE23535
fixed: m:vvebedeeefeg
uid: h:31323334350a
key: h:05311f79fc8d7b6cdf10abf86920203b      <-- Dieser Wert ist notwendig (nach h:)
acc_code: h:000000000000
ticket_flags: APPEND_CR
config_flags:
```

Wir prüfen dann also den Key mit

```
ykdebug 05311f79fc8d7b6cdf10abf86920203b vvebedeeefeglgltbffvvjrghveitcvertrdltnuhecv
```

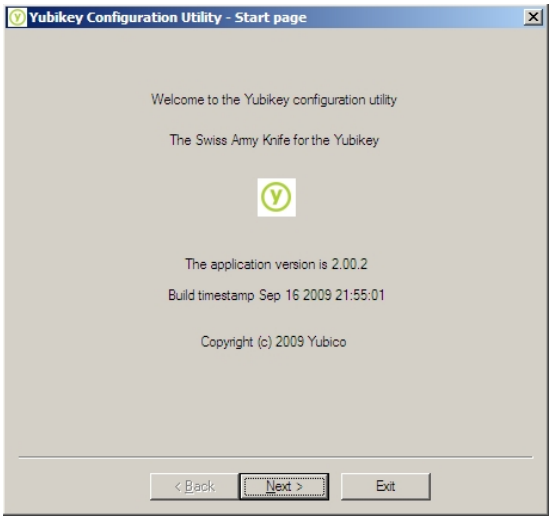
und das Ergebnis sollte in etwa so aussehen:

```
warning: overlong token, ignoring prefix: vvebedeeefeg
Input:
  token: lgltbffvvjrghveitcvertrdltnuhecv
         a5 ad 14 4f f8 c5 6f 37 d0 f3 cd c2 ad be 63 0f
  aeskey: 05311f79fc8d7b6cdf10abf86920203b
         05 31 1f 79 fc 8d 7b 6c df 10 ab f8 69 20 20 3b
Output:
         31 32 33 34 35 0a 02 00 ac 3e 5c 01 f3 94 b1 a1

Struct:
  uid: 31 32 33 34 35 0a
  counter: 2 (0x0002)
  timestamp (low): 16044 (0x3eac)
  timestamp (high): 92 (0x5c)
  session use: 1 (0x01)
  random: 38131 (0x94f3)
  crc: 41393 (0xa1b1)

Derived:
  cleaned counter: 2 (0x0002)
  modhex uid: ebedeeefegcl
  triggered by caps lock: no
  crc: F0B8
  crc check: ok
```

Wie wir sehen war alles in Ordnung. Der Yubikey wurde erfolgreich mit einem uns bekannten AES-Key reprogrammiert.

AES Key reprogrammieren [Windows]

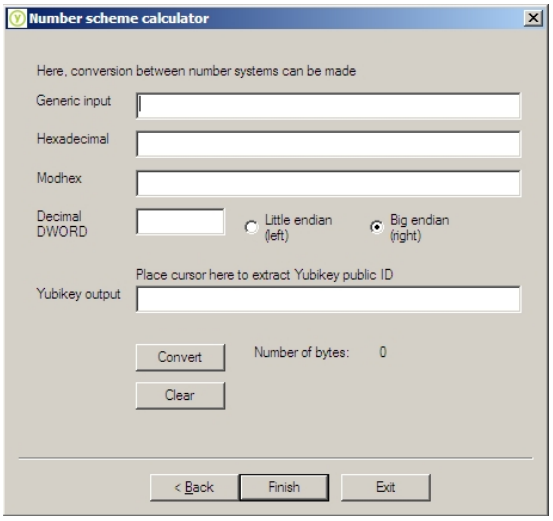
Zunächst laden wir uns das Personalization Tool für Windows von http://www.yubico.com/developers/personalization/ [http://www.yubico.com/developers/personalization/]herunter und installieren es.
ACHTUNG - AB HIER NICHT ABSCHREIBEN, SONDERN SELBST MACHEN. ALLE WERTE UND SCREENSHOTS HIER SIND UNBRAUCHBAR UND DIENEN NUR ZUR VERANSCHAULICHUNG!
Als administrativer User startet wir nun YubikeyConfig.exe es und sehen folgenden Bildschirm:



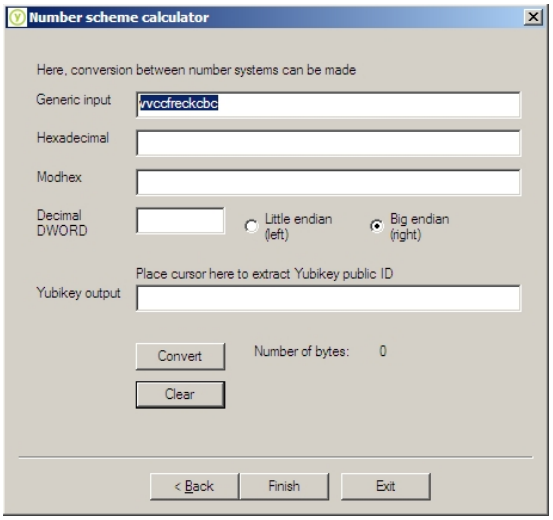
Durch Klicken auf **Next** gelangen wir in die Software. Wir benötigen zunächst den Punkt

```
[x] convert between different number formats
```

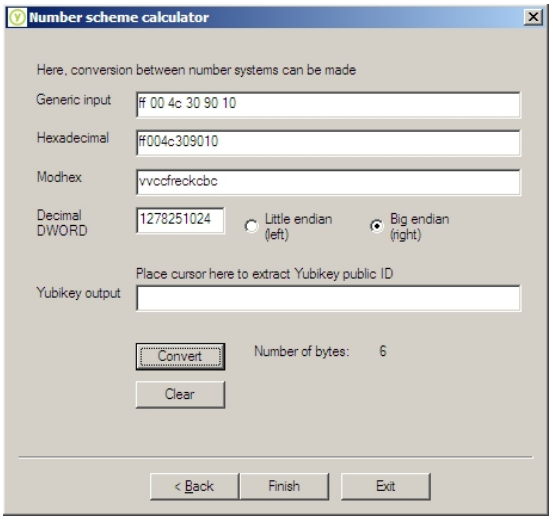
um unseren zukünftigen **AES-Key** und unseren **private identity string** zu ermitteln. Dort sehen wir nach Klicken auf **Next** folgenden Bildschirm:



Wir geben bei **Generic input** zunächst einen Ubikey Prefix ein, der **unbedingt** mit **vv** (2x v) beginnen muss und insgesamt (inklusive vv) maximal 12 Zeichen lang ist.



Dann drücken wir auf **Convert** und erhalten folgendes Ergebnis.



Wir notieren uns die Daten als

Generic	Hex	ModHex	Dec
ff 00 4c 30 90 13	ff004c309013	vvccfreckcbe	1278251024

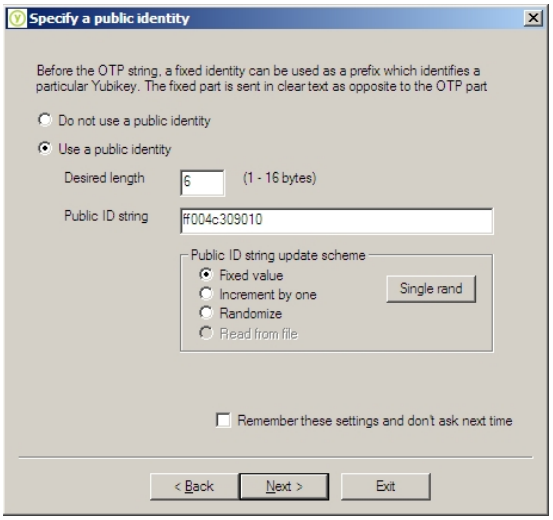
Dieser Wert wird später die **public identity** Kennung für unseren Yubikey (die ersten 12 Zeichen = 6 Byte) sein.

Nun wechseln wir wieder zurück auf den Hauptbildschirm

und wählen

```
[x] Create a dynamic Yubikey configuration (OTP mode)
```

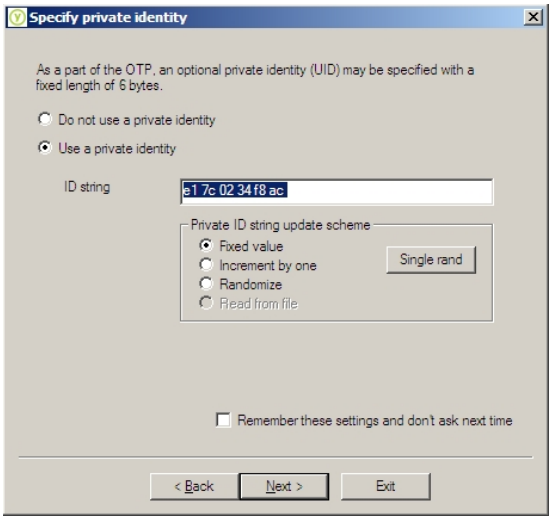
Im nächsten Bildschirm wählen wir [x] Use a public identity und geben als **Desired length** 6 und als Public ID string den Hex-Wert ein, den wir oben ermittelt haben, z.B. **ff004c309013**



Dann klicken wir auf **Next**. Hier wählen wir

```
[x] Use a private identity
```

und **Fixed value** und drücken auf **Single rand**. Dadurch wird ein zufälliger String ermittelt.



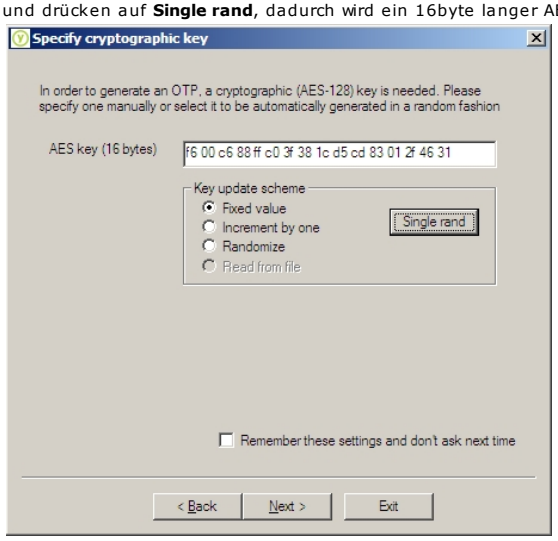
Wir notieren uns die Daten als

Private Identity-ID-String-Hex
e1 7c 02 34 f8 ac

nun klicken wir wieder auf **Next** und wählen im nächsten Bildschirm

```
[x] Fixed value
```

und drücken auf **Single rand**, dadurch wird ein 16byte langer AES-Key generiert:



Diesen notieren wir uns wieder als

AES key-Hex
f6 00 c6 88 ff c0 3f 38 1c d5 cd 83 01 2f 46 31

und drücken **Next**

Im nächsten Bildschirm kann man die Output-Parameter des UbiKeys festlegen. Der Standard ist

```
[x] Send ENTER as the last keystroke
```

Wenn man andere Einstellungen benötigt, hier bitte Vorsicht, da einige Einstellungen nicht verwendet werden dürfen.
Siehe dazu auch den User Guide [http://www.yubico.com/files/AES%20Key%20Upload%20Guide%202009-09-15.pdf]. Ansonsten drücken wir **Next**

Der nächste Bildschirm erlaubt es uns, die Konfiguration des Yubikeys per Passwort zu schützen. Dies ist optional, kann jedoch für Firmen, die reprogrammierte Yubikeys an ihre Mitarbeiter ausgeben sinnvoll sein, damit die Mitarbeiter nicht selbst versuchen, die Yubikeys umzuprogrammieren. Für unser Beispiel lassen wir den Standard:

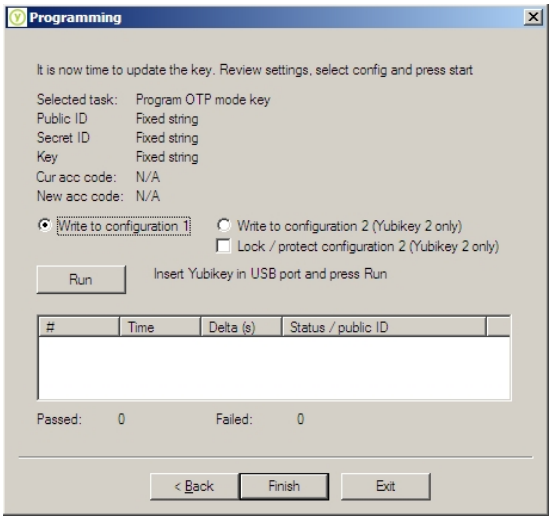
```
[x] The Yubikey(s) are currently unprotected and I want to keep it that way
```

und drücken **Next**

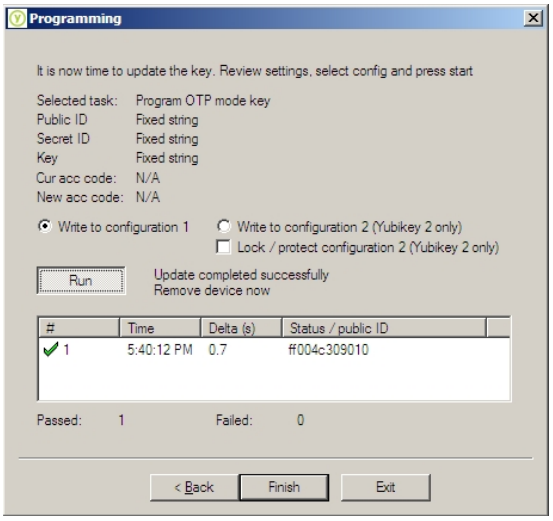
Im letzten Bildschirm kann man nun den Ort der Konfiguration wählen. Dazu empfehle ich auch die Dokumentation zu lesen.Wenn man in die **configuration 2 (Yubikey 2 only)** schreibt, dann verändert man nichts an der in der **configuration 1** gespeicherten Auslieferungszustands.
Da wir später unseren AES-Key zu Yubico hochladen werden und somit auch die Konfiguration 1 wieder voll funktionsfähig sein wird, wählen wir hier:

```
[x] Write to configuration 1
```

und drücken auf den Button **Run**.



Das Resultat sollte in etwa folgendermaßen aussehen:



Mit **Finish** beenden wir das Tool.

Wenn wir den Yubikey nun nutzen, sollte ungefähr so ein String rauskommen (die ersten 12 Zeichen sollten dem ModHex Wert unseres Public Strings entsprechen)

```
vvccfreckcbebhvkcturhgtjubrnkvcujhelrfebrfgh
```

Der Key wäre nun reprogrammiert und wir könnten ihn bereits mit obigem PAM-Modul verwenden. Da wir jedoch die **configuration 1** überschrieben haben, würde der Key nie wieder bei Online-Diensten funktionieren, die als Backend die Authentifizierungsserver von Yubico verwenden, deshalb teilen wir Yubico noch eben per WebInterface die neuen Daten unseres Keys mit.

Benutzer mit Yubikey assoziieren in /etc/yubikey

Um nun einen Benutzer zur /etc/yubikey hinzuzufügen (vergleichbar mit /etc/passwd), wird das Tool **ykpasswd** benötigt. Dieses haben wir durch die Installation von **YubiPAM** bereits auf das System gebracht.
Für die Zuweisung eines UbiKeys benötigen wir folgende Angaben:

- AES-Key in HEX
- fixed public name in ModHex
- uid in HEX

Wenn wir die Programmierung unter Linux mit "yubiprogramprepare" durchgeführt haben, sollten alle benötigten Daten in der abgespeicherten Datei (z.B. FILE23535) zu finden sein.

```
cat FILE23535
```

```
fixed: m:vvebedeeefeg                    <-- fixed public name (nach m:)
uid: h:31323334350a                      <-- uid               (nach h:)
key: h:05311f79fc8d7b6cdf10abf86920203b  <-- AES-Key           (nach h:)
acc_code: h:000000000000
ticket_flags: APPEND_CR
config_flags:
```

Angenommen wir wollen für den user **root** den UbiKey für den die obigen Daten zutreffen zur Authentifizierung nutzen, dann geben wir folgenden Befehl ein:

```
ykpasswd -a -k 05311f79fc8d7b6cdf10abf86920203b -f vvebedeeefeg -p 31323334350a root
```

Als Ausgabe sollte dann erscheinen:

```
Adding Yubikey entry for root
Completed successfully.
```

Wenn man die Werte nach der Reihe eingeben will (prompt), dann reicht auch ein

```
ykpasswd -a root
```

Die Überprüfung des UbiKeys erfolgt mit dem Tool **ykvalidate** welches ebenso per YubiPAM bereits installiert ist:

```
ykvalidate -u root OTP
```

wobei anstatt OTP 1x auf den UbiKey gedrückt wird. Das Ergebnis sollte dann so aussehen:

```
ykvalidate -u root vvebedeeefegkkrlcrunlfgbrjedkiukvhbbfielclhc
OTP is VALID.
```

SSH logins für PAM mit YubiPAM konfigurieren

Wie man PAM mit YubiPAM realisiert steht detailiert in der README

```
/usr/share/doc/yubipam/README
```

Nur wenn wir bereits erfolgreich einen User zu /etc/yubikey hinzugefügt haben und die erfolgreiche Authentifizierung überprüft haben, editieren wir die Datei **/etc/pam.d/sshd**

```
vi /etc/pam.d/sshd
```

und fügen ganz oben folgende Zeile ein.

```
auth sufficient pam_yubikey.so
```

oder

```
auth sufficient pam_yubikey.so verbose_otp
```

Der Parameter **verbose_otp** ist optional und sollte nur zu Testzwecken eingefügt werden. Dadurch wird der komplette Passwort-Auswertevorgang nach stdin ausgegeben - und wir **überprüfen noch mal sicherheitshalber, ob pam_yubikey.so im Ordner /lib/security existiert.**

```
ls -l /lib/security/pam_yubikey.so
/lib/security/pam_yubikey.so
```

SSH Daemon Konfiguration anpassen

Um sich per UbiKey am SSH Server anmelden zu können, muss in der Konfigurationsdatei

```
/etc/ssh/sshd_config
```

die Optionen

```
PasswordAuthentication yes
```

und

```
ChallengeResponseAuthentication yes
```

gesetzt sein.

Ist

```
ChallengeResponseAuthentication no
```

gesetzt, erhält man **keinen speziellen "Yubikey OTP:" Prompt**.

Sind beide Optionen gesetzt, läuft ein Anmeldung per SSH, nach einem Restart des SSH-Daemons:

```
/etc/init.d/ssh restart
```

folgendermaßen ab:

```
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is 9e:b3:f6:ab:a1:9f:8f:03:64:de:16:64:84:0e:4f:f7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
Yubikey OTP: vvebedeeefeglekkgunfbncgbultjbhdglengivkrugn

Linux lenny-otp 2.6.26-2-686 #1 SMP Wed Nov 4 20:45:37 UTC 2009 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Jan 12 18:53:03 2010 from 192.168.178.97
lenny-otp:~#
```

Hier noch zur Sicherheit meine **/etc/ssh/sshd_config** ohne Kommentare und Leerzeichen

```
cat /etc/ssh/sshd_config | grep -E -v "^($|#)" | sed -e 's/^\(.*\)$/  \1/g'
```

/etc/ssh/sshd_config

```
Port 22
Protocol 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
UsePrivilegeSeparation yes
KeyRegenerationInterval 3600
ServerKeyBits 768
SyslogFacility AUTH
LogLevel INFO
LoginGraceTime 120
PermitRootLogin yes
StrictModes yes
RSAAuthentication yes
PubkeyAuthentication yes
IgnoreRhosts yes
RhostsRSAAuthentication no
HostbasedAuthentication no
PermitEmptyPasswords no
 ChallengeResponseAuthentication yes
 PasswordAuthentication yes
X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
AcceptEnv LANG LC_*
Subsystem sftp /usr/lib/openssh/sftp-server
UsePAM yes
```

Geänderten Key uploaden

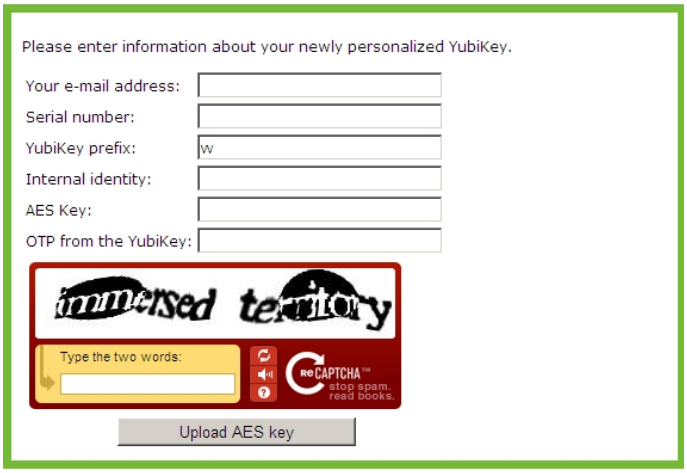
Wenn wir die Programmierung unter Linux mit "yubiprogramprepare" durchgeführt haben, sollten alle benötigten Daten in der abgespeicherten Datei (z.B. FILE23535) zu finden sein.

```
cat FILE23535
```

```
fixed: m:vvebedeeefeg                    <-- fixed public name (nach m:)
uid: h:31323334350a                      <-- uid               (nach h:)
key: h:05311f79fc8d7b6cdf10abf86920203b  <-- AES-Key           (nach h:)
acc_code: h:000000000000
ticket_flags: APPEND_CR
config_flags:
```

Wenn der Schlüssel unter Windows reprogrammiert wurde, sollten sie die Daten aufgezeichnet haben (z.B. Screenshots) Wir besuchen die Seite: Yubico AES Key Upload [https://api.yubico.com/upload-aes-key/]

Dort erscheint dann folgende Eingabemaske:



Wir füllen folgende Daten ein:

Your e-mail address:	ihre@email.de	
Serial number:	0036839	von Verpackung, ansonsten geht auch eine 0
YubiKey prefix:	vvebedeeefeg	ModHex Wert public ID
Internal identity:	31323334350a	Private Identitiy-ID-String in Hex
AES-Key:	05311f79fc8d7b6cdf10abf86920203b	AES Key in Hex
OTP from the YubiKey:	vvebedeeefegglnictrivtnkirjteujndnltgvthfkkl	(erst wenn Captcha ausgefüllt)

Das Ergebnis sollte erfolgreich sein und in etwas so aussehen:

Success! Key upload successful.

E-mail address:	ihre@email.de
Serial number:	0036839
Yubikey prefix:	vvebedeeefeg
Internal identity:	31323334350a
AES key:	05311f79fc8d7b6cdf10abf86920203b
YubiKey OTP:	vvebedeeefegglnictrivtnkirjteujndnltgvthfkkl

Nun warten wir ca. 10 Minuten.
BITTE NICHT WEITERMACHEN BEVOR DIE 10 MINUTEN VERSTRICHEN SIND, DA ES SONST NICHT FUNKTIONIERT DER NEUE KEY MUSS ERST BEI YUBICO INTERN VERARBEITET SEIN, BEVOR WIR IHN ONLINE PRÜFEN KÖNNEN UND DAS DAUERT EIN WENIG!

Danach können wir kurz prüfen, ob unsere UbiKey (nach Reprogrammierung) auch noch für Online-Authentifizierung durch Yubico geeignet ist.

Dazu gehen wir auf folgende Seite und testen den Key.
http://www.yubico.com/1
Das Ergebnis sollte sein:

```
Demo YubiKey only
Congratulations
You have been successfully authenticated with the YubiKey.
```

LINKS

http://code.google.com/p/yubikey-personalization/ [http://code.google.com/p/yubikey-personalization/]
yubikey-personalization
https://admin.yubico.com/yubirevoke/login.php [https://admin.yubico.com/yubirevoke/login.php]
Dort kann man auch den AES Key bekommen und zwar ohne Re-Programmierung
http://code.google.com/p/yubico-c-client/ [http://code.google.com/p/yubico-c-client/]
yubico-c-client
http://code.google.com/p/yubico-pam/ [http://code.google.com/p/yubico-pam/]
yubico-pam
http://code.google.com/p/yubico-c/ [http://code.google.com/p/yubico-c/]
libyubikey
http://www.linuxjournal.com/article/10166 [http://www.linuxjournal.com/article/10166]
Yubikey One-Time Password Authentication
http://forum.yubico.com/viewtopic.php?f=6&t=96 [http://forum.yubico.com/viewtopic.php?f=6&t=96]
About ModHex
http://www.yubico.com/demo/Modhex_Calculator.php [http://www.yubico.com/demo/Modhex_Calculator.php]
Yubico ModHex Calculator
https://wiki.duckcorp.org/YubikeyHelp [https://wiki.duckcorp.org/YubikeyHelp]
Sehr gute Anleitung zu Yubikeys unter Linux
http://en.wikipedia.org/wiki/AAA_protocol [http://en.wikipedia.org/wiki/AAA_protocol]
AAA Protokoll
http://www.yubico.com/demo/one_factor.php [http://www.yubico.com/demo/one_factor.php]
Online Prüfung Yubikey
https://api.yubico.com/upload-aes-key/ [https://api.yubico.com/upload-aes-key/]
AES Key Upload zu Yubico
https://admin.yubico.com/yubirevoke/login.php [https://admin.yubico.com/yubirevoke/login.php]
Yubico Revoke Service
YubiPAM Project Page [http://www.securixlive.com/yubipam/index.php]
Yubikey Authentication on Linux [http://blog.rootshell.be/2009/03/27/yubikey-authentication-on-linux/]
yubidus - A radius server designed to be used with wpa/wpa2 enterprise networks and Yubikeys [http://code.google.com/p/yubidus/source/detail?r=7]
YubiRADIUS Service by Yubico [http://wiki.yubico.com/wiki/index.php/Applications:YubiRADIUS_Hosted_RADIUS_Service]
yubico-pam - Yubico Pluggable Authentication Module (PAM) [http://code.google.com/p/yubico-pam/]
freeradius-yubikey-module [http://code.google.com/p/freeradius-yubikey-module/]
Yubikey Forum [http://forum.yubico.com]
Modhex Calculator [http://www.yubico.com/demo/Modhex_Calculator.php]

---

[1] YubiPAM muss vorher kompiliert werden, siehe entspr. Punkt in dieser Anleitung.

[2] Die Reprogrammierung des Yubikeys unter Linux erfordert das Vorhandensein von libyubikey und ykpers. Beide müssen vorher kompiliert werden, siehe entspr. Punkt in dieser Anleitung.