

Technical Guide about:

**OpenSSL user authentication with
Apache using x.509 certificates on
smart cards**

by Daniel Struck

(elaborated with the help of: Stef Hoeben, Olaf Kirch & Nils Larsch)

Table of Contents

1	Introduction.....	3
2	Public Key Infrastructure (PKI).....	4
2.1	Smart cards.....	4
2.1.1	Installation of the smart cards on Windows.....	4
2.1.2	Installation of the smart cards on Linux.....	4
2.2	Creation of a Certificate Authority (CA).....	5
2.2.1	Creation of an environment for the CA.....	5
2.2.2	Creation of the CA:.....	6
2.2.3	Install the CA certificate as an authority in Mozilla.....	6
2.3	Creation of certificates for the users.....	6
2.4	Create a certificate for the Web server.....	7
2.4.1	Configuration of Apache2.....	8
2.5	Management of the user certificates.....	9
2.5.1	Revoke a user certificate:.....	9
2.5.2	Replace an expired user certificate on a smart card.....	9
2.5.3	Unblock a smart card.....	10
2.6	S/MIME: sign & encrypt mails with Mozilla Thunderbird.....	10
3	Appendix.....	11
3.1	Useful commands for openssl:.....	11
3.2	Abbreviations.....	12
3.3	openssl.cnf.....	13

Version 0.5 (25.11.2003):

- adaptation to changes made in the profile management of openssl

Version 0.4 (13.10.2003):

- added S/MIME section
- changed the key usage to digital signature

Version 0.3 (04.10.2003):

- use flex_so.profile instead of the default profile ("flex.profile")

Version 0.2 (30.09.2003):

- updates based on remarks from Stef Hoeben

Version 0.1 (29.09.2003):

- initial draft

1 Introduction

Purpose: Authenticate users of a web application with smart cards.

Hardware: smart cards supported by a pkcs15-driver from OpenSC¹

I have tested the configuration with e-gate cryptoflex smart cards from Schlumberger² used together with their token connectors³.

Software: Apache on Windows⁴ or Linux

(configuration has been tested on Gentoo-Linux with Apache 2.0.48-r1)

Client browsing the content with Mozilla on Windows or Linux

(configuration has been tested with Mozilla 1.4 & MozillaFirebird 0.7⁵ on WindowsXP and Linux)

Nowadays a lot of content on Internet or intranet is served by a web application based on the LAMP-platform (Linux Apache MySQL PHP). In some cases this content consists of sensitive data, which should be secured appropriately. Often the access to the web application is protected by one of these methods:

- Apache basic authentication
- php login scripts (self-developed or a pear package)
- ...

The big disadvantage of these methods, is the usage of passwords for the user authentication. Users often write their personal passwords in their agenda or on a post-it , which sticks directly on the monitor.

With the help of the OpenCT and the OpenSC it is possible to protect the access to Apache with the help of smart cards.

1 <http://www.opensc.org/>

2 <http://www.reflexreaders.com/Products/e-gate/e-gate.html>

3 http://www.scmegastore.com/cgi-bin/webc.cgi/st_prod.html?p_prodid=112

4 For an easy setup LAMP system for Windows, have a look at <http://www.opensa.org/> or <http://www.apachefriends.org/>

5 You need to install the "Things They Left Out" extension, to gain access to the security manager (not needed anymore in Firebird 0.7)

2 Public Key Infrastructure (PKI)

To perform authentication of clients we need to install a small Public Key Infrastructure (PKI)⁶.

2.1 Smart cards

First let us install the smart card drivers and tools under Linux and / or Windows, to be able to create the PKI infrastructure and to prepare the smart cards for the clients.

2.1.1 Installation of the smart cards on Windows

Install the original drivers from Schlumberger⁷.

To access your smart card in Mozilla, you must install a PKCS#11 device. You will find binary drivers from OpenSC at "<http://www.opensc.org/files/>":

- Extract the Zip-File to "<c:\opensc>".
- Install the following DLL's to "c:\windows\system32": libeay32.dll & opensc.dll
- Install the PKCS#11 Device in Mozilla
(Edit > Preferences > Privacy & Security > Certificates > Manage Security Devices > Load > [browse to <c:\opensc> and select the module "opensc-pkcs11.dll"])
- you will now be able to use the smart card in Mozilla.

Note: This documentation is based on the latest snapshot, which has a different profile management as the 0.8.1 release.

Until new binary drivers (>0.8.1) are released, you should use the latest snapshot and compile the drivers yourself for Windows.

2.1.2 Installation of the smart cards on Linux

A laptop running under Linux will be used to prepare the smart cards for the clients. Of course you could also perform the same on Windows, but I have only tested it on Linux.

First we need to install the drivers for the e-gate cryptoflex smart card. For

⁶ "Network Security with OpenSSL" by Pravar Chandra, Matt Messier, John Viega (O'Reilly) In this book you will find basic explanations how to establish a basic PKI with OpenSSL

⁷ <http://www.reflexreaders.com/Support/Downloads/downloads.html>

this purpose, we will be using the OpenCT package. Within the tar-file of OpenCT, you will find detailed install instructions.

Be aware that you could also use the drivers from the M.U.S.C.L.E project⁸ for your smart card.

OpenSC will provide all the tools necessary to prepare the smart cards for the clients. As for OpenCT, you will find detailed install instructions in the tar-file.

Compile OpenSC with support for OpenCT:

```
cd [path to OpenSC sources]
./configure --prefix=/usr --sysconfdir=/etc --with-
opentct=/usr
make
su
make install
```

Attention you must manually install “opensc.conf”, or else the OpenSC-engine won't work in OpenSSL:

```
cp /usr/share/opensc/opensc.conf.example \
/etc/opensc.conf
```

Note: for this step by step guide I have used the latest snapshot (opentct-20031210.tar.gz & opensc-20031210.tar.gz)

2.2 Creation of a Certificate Authority (CA)

First we must create the keys for the CA. The CA is responsible to sign the certificates of the clients.

2.2.1 Creation of an environment for the CA

In our case, we will setup an environment for the CA in the directory “/etc/ssl/test”:

```
mkdir certs
touch index.txt
echo "01" > serial
```

serial: _____ contains the serial number attributed to the client-certificate by

8 <http://www.linuxnet.com/>

the CA.

index.txt: sort of database, that keeps track of the client-certificates issued by the CA. This index will be useful later, if you need to revoke a certificate.

For the configuration file of OpenSSL see the Appendix (openssl.conf).

2.2.2 Creation of the CA:

Steps required to create a CA:

```
pkcs15-init -EC -T (Erase smart card and create an initial
pkcs15 structure)

pkcs15-init -P -a 01 -T (create a container "01" with PIN &
PUK)

pkcs15-init -G rsa/2048 -a 01 --key-usage digitalSignature
(generate a key pair for the Retro CA in container "01", we
will be using a key length of 2048bit for the CA)

openssl>engine dynamic -pre
SO_PATH:/usr/lib/opensc/engine_opensc.so -pre ID:opensc -pre
LIST_ADD:1 -pre LOAD(load the opensc-engine in openssl)

openssl> req -engine opensc -new -x509 -days 730 -key 45
-keyform engine -out /etc/ssl/test/cacert.pem (create CA
certificate and self-sign it, the cert will be valid for 2
years)

pkcs15-init --store-certificate cacert.pem --authority
(store certificate on smart card, not necessary for the CA)
```

2.2.3 Install the CA certificate as an authority in Mozilla

You need to install the CA certificate in Mozilla as an authority to be able to access the protected website:

- In Mozilla go to “Edit > Privacy Security > Certificates > Manage Certificates > Authorities”.
- Import the CA certificate “cacert.pem”.

2.3 Creation of certificates for the users

We will prepare a smart card for a user, by issuing the following commands:

```
pkcs15-init -EC -T (Erase smart card and create a initial
pkcs15 structure)

pkcs15-init -P -a 01 -T (create a container "01" with PIN &
PUK)

pkcs15-init -G rsa/1024 -a 01 --key-usage digitalSignature
(if you want to use the key also for mail encryption, do add
"keyEncipherment" to the key usage9)

openssl>engine dynamic -pre
SO_PATH:/usr/local/lib/opensc/engine_opensc.so -pre
ID:opensc -pre NO_VCHECK:1 -pre LIST_ADD:1 -pre LOAD

openssl>req -engine opensc -new -key 45 -keyform engine -out
[user name].pem

(eject user smartcard and insert CA smartcard)

openssl>engine dynamic -pre
SO_PATH:/usr/local/lib/opensc/engine_opensc.so -pre
ID:opensc -pre NO_VCHECK:1 -pre LIST_ADD:1 -pre LOAD

openssl>ca -engine opensc -keyfile 45 -keyform engine -in
[user name].pem (CA signs the user certificate)

(eject CA smartcard and insert user smartcard)

pkcs15-init --store-certificate [user name].pem -a 01

(store user certificate on smartcard)
```

2.4 Create a certificate for the Web server

Apache does need a certificate to establish SSL-connections. These are the steps required to create this certificate:

```
openssl req -newkey rsa:1024 -nodes -keyout server.key
-keyform PEM -out server.csr (Create a certificate request)

(Attention, the option "-nodes", which means "use no pass
phrase for the private key", should only be used on a
development platform!)

(Also make attention to use the Server Name in the "Common
Name" Field for the certificate request. Example:
"localhost" for a development platform, where the web server
is running as localhost.)

(insert the CA smart card)

openssl>engine dynamic -pre
SO_PATH:/usr/lib/opensc/engine_opensc.so -pre ID:opensc -pre
NO_VCHECK:1 -pre LIST_ADD:1 -pre LOAD
```

- 9 The combination of digitalSignature and keyEncipherment is not supported by all card operating systems.
It does work on the e-gate cryptoflex smart cards.

```
openssl>ca -engine openssl -keyfile 45 -keyform engine -name
CA_server -in server.csr -out server.crt (sign certificate
with CA)
```

Attention: Do not create a certificate for the web server with the “01” serial, or else Mozilla will complain, that a certificate with that serial number already exists. So create first a user certificate, then the web server certificate.

2.4.1 Configuration of Apache2

Next we must configure Apache2 to use the web server certificate and the CA certificate to authenticate the users:

/etc/apache2/conf/apache2.conf

```
# don't use SSLv2, as it has fundamental design problems
SSLProtocol all -SSLv2

# all ciphers using strong encryption (Triple-DES)
SSLCipherSuite HIGH

## Deny access when SSL is not used for the HTTP request
## Will be activated when the Database Server is functional
#SSLRequireSSL

## require a client certificate which has to be directly
## signed by our CA certificate in ca.crt
SSLVerifyClient require
SSLVerifyDepth 1

# depth of 1 means the client certificate has to be signed by a CA which is directly known to
# the server
SSLCACertificateFile /etc/ssl/test/cacert.pem

## Location of the Webserver Certificate & private key
SSLCertificateKeyFile /etc/ssl/test/server.key
SSLCertificateFile /etc/ssl/test/server.crt

## Location of the Revocation List
SSLCACertificateFile /etc/ssl/test/test.crl

## Export environment variables for php
SSLOptions +StdEnvVars
```

After the configuration, restart Apache2, on Gentoo you can do it by issuing the command:

```
/etc/init.d/apache2 reload
```

2.5 Management of the user certificates

2.5.1 Revoke a user certificate:

In case a user loses his smart card or is not sure his PIN is safe any more, you must revoke the client certificate:

- Localize the user certificate “/etc/ssl/retro/certs/xy.pem” in the file-database “/etc/ssl/retro/index.txt”.
- Revoke the user certificate:

```
openssl ca -revoke [xy.pem]
```

- Create a Certificate Revocation List (CRL):

```
openssl ca -gencrl -out test.crl
```

2.5.2 Replace an expired user certificate on a smart card

The certificates of the users in this configuration are valid for one year. After they expire, you need to delete on the smart card and replace it by a new valid certificate.

Steps required to delete a certificate on a smart card:

- Determine the path to the certificate on the smart card

```
pkcs15-tool -c
> Connecting to card in reader Schlumberger E-Gate...
> Using card driver: Schlumberger Multiflex/Cryptoflex
> Trying to find a PKCS#15 compatible card...
> Found smartcard used for a user certificate!
> Card has 1 certificate(s).

> X.509 Certificate [Certificate]
```

```
>      Flags      : 2
>      Authority: no
>      Path       : 3F0050155501
>      ID        : 45
```

(the path "5015" is marked in bold, 5501 is the certificate)

- Use "opensc-explorer to delete the certificate and the CDF of the certificate:

```
opensc-explorer> verify AUT1 [transport key]
(transport key is in our case: 2c:15:e5:26:e9:3e:8a:19)
opensc-explorer>cd 5015
opensc-explorer>delete 5501  (delete certificate)
opensc-explorer>delete 4404  (delete the CDF, the
                             certificate directory file)
```

2.5.3 Unblock a smart card

If a users enters 3 times a wrong PIN, the smart card will be blocked. Once a PIN is blocked you can present any PIN, even the correct one, but you will never get access the the protected key pair. The smart card then can only be unblocked with the corresponding PUK:

```
pkcs15-tool -u -a 01
```

2.6S/MIME: sign & encrypt mails with Mozilla Thunderbird

A nice bonus of the generated certificates on the smart cards, is the fact that you can use them for S/MIME on Mozilla or on Mozilla Thunderbird.

As Mozilla Thunderbird is going to be the future email program of Mozilla, I am going to describe here the steps to setup this email program for S/MIME (the steps for the integrated email program of Mozilla should be similar):

- Install the PKCS#11 Device in Mozilla Thunderbird
(Tools > Account settings > [your email account to be used] > Security > Manage Security Devices > Load > [browse to <c:\opensc> and select the module "opensc-pkcs11.dll"])
- Install the Certificate Authority
(Tools > Account settings > [your email account to be used] > Security > Manage Certificates > Authorities > import > [browse to "cacert.pem" and

select it)

- Insert your smart card and select the certificates to be used for “Digital Signing” and “Encryption of your emails

(Tools > Account settings > [your email account to be used] > Security > Select ...)

3Appendix

3.1Useful commands for openssl:

To read the content of a certificate, issue the command:

```
openssl x509 -in xyz.crt -noout -text
```

To print out information about a certificate revocation list:

```
openssl crl -in xyz.crl -noout -text
```

3.2 Abbreviations

CDF	Certificate Directory File, a pkcs15 file that contains info (label, ID, path, ...) of the certs. It is divided into entries, one entry per cert.
DER	"Distinguished Encoding Rules". A binary format x.509 certificates are represented in.
PIN	Password protecting a private key on a smart card
PUK	Password to unblock a PIN.

3.3 openssl.cnf

```
#
# OpenSSL configuration file for the Test CA.
#

# This definition stops the following lines choking if HOME isn't
# defined.
HOME          = .
RANDFILE      = $ENV::HOME/.rnd

# Extra OBJECT IDENTIFIER info:
oid_file      = $ENV::HOME/.oid
oid_section   = new_oids

# To use this configuration file with the "-extfile" option of the
# "openssl x509" utility, name here the section containing the
# X.509v3 extensions to use:
# extensions  =

# (Alternatively, use a configuration file that has only
# X.509v3 extensions in its main [= default] section.)

[ new_oids ]

# We can add new OIDs in here for use by 'ca' and 'req'.
# Add a simple OID like this:
# testoid1=1.2.3.4
# Or use config file substitution like this:
# testoid2=${testoid1}.5.6

#####

[ ca ]
default_ca = CA_default          # The default ca section
#####

[ CA_default ]

dir                = /etc/ssl/test          # Where everything is kept
certificate        = $dir/cacert.pem       # The CA certificate
database           = $dir/index.txt       # database index file.
new_certs_dir      = $dir/certs           # default place for new certs.
```

```

serial          = $dir/serial          # The current serial number

x509_extensions = usr_cert             # The extensions to add to the cert

# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
# crl_extensions = crl_ext

default_days    = 365                  # how long to certify for
default_crl_days = 30                  # how long before next CRL
default_md = sha1                       # which md to use (md5,sha1,mdc2).
                                           # will try out if sha1 works with mozilla browser & email
                                           # then also set it in [req] !!!!!!!!!!!

preserve        = no                   # keep passed DN ordering

# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy          = policy_match

[ CA_server ]

dir             = /etc/ssl/test         # Where everything is kept
certificate     = $dir/cacert.pem      # The CA certificate
database        = $dir/index.txt      # database index file.
new_certs_dir   = $dir/certs           # default place for new certs.
private_key     = $dir/private/cakey.pem # The private key
                                           # isn't really needed as we put
                                           # the private key on a smartcard

serial          = $dir/serial          # The current serial number

x509_extensions = server_cert         # The extensions to add to the cert
default_days    = 365                  # how long to certify for
default_crl_days = 30                  # how long before next CRL
default_md = sha1                       # which md to use.
Preserve        = no                   # keep passed DN ordering
policy          = policy_match

# For the CA policy
[ policy_match ]
countryName     = match

```

#stateOrProvinceName = optional
organizationName = match
organizationalUnitName = match
commonName = supplied
emailAddress = supplied

#####

[req]

default_bits = 2048
default_keyfile = *privkey.pem*
default_md = *sha1* # which md to use.

distinguished_name = *req_distinguished_name*
#attributes = *req_attributes*
x509_extensions = *v3_ca* # The extensions to add to the self signed cert

This sets a mask for permitted string types. There are several options.
default: PrintableString, T61String, BMPString.
pkix : PrintableString, BMPString.
utf8only: only UTF8Strings.
nombstr: PrintableString, T61String (no BMPStrings or UTF8Strings).
MASK:XXXX a literal mask value.
WARNING: current versions of Netscape crash on BMPStrings or UTF8Strings
so use this option with caution!
string_mask = *nombstr*

[req_distinguished_name]

countryName = Country Name (2 letter code)
countryName_default = LU
countryName_min = 2
countryName_max = 2

localityName = Locality Name (eg, city)
localityName_default = Luxembourg

organizationName = Organization Name (eg, company)
organizationName_default = Test-Organization

organizationalUnitName = Organizational Unit Name (eg, section)
organizationalUnitName_default = Test-Unit

commonName = Common Name (eg, YOUR name)

commonName_max = 64

emailAddress = Email Address

emailAddress_max = 40

emailAddress_default = admin@test.lu

[*usr_cert*]

These extensions are added when 'ca' signs a request.

This goes against PKIX guidelines but some CAs do it and some software

requires this to avoid interpreting an end user certificate as a CA.

basicConstraints=CA:FALSE

For normal client use this is typical

nsCertType = client, email

This is typical in *keyUsage* for a client certificate.

keyUsage = digitalSignature, keyEncipherment

This will be displayed in Netscape's comment listbox.

nsComment = "OpenSSL Generated Certificate for a Test user"

don't use any of the following options for a user certificate to be put on a smartcard !!!

else it doesn't work in mozilla-browser

#*subjectKeyIdentifier=hash*

#*authorityKeyIdentifier=keyid,issuer:always*

#*subjectAltName=email:copy*

#*issuerAltName=issuer:copy*

[*server_cert*]

basicConstraints = CA:FALSE

nsCertType = server

keyUsage = keyEncipherment

nsComment = "OpenSSL Generated Certificate for a Webserver"

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid,issuer:always

subjectAltName = email:copy

issuerAltName = *issuer.copy*

[*v3_ca*]

Extensions for a typical CA

PKIX recommendation.

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer:always

This is what PKIX recommends but some broken software chokes on critical

extensions.

basicConstraints = CA:true

Key usage: this is typical for a CA certificate. However since it will

prevent it being used as an test self-signed certificate it is best

left out by default.

keyUsage = cRLSign, keyCertSign

Some might want this also

Moeglich: client, server, email, objsign, reserved, sslCA, emailCA, objCA

nsCertType = sslCA, emailCA

Include email address in subject alt name: another PKIX recommendation

subjectAltName=email.copy

Copy issuer details

issuerAltName=issuer.copy