The OpenCA Guide

Michael Bell, Chris Covell and Harald Wallus

Email: michael.bell@web.de, chris@katjam.co.uk, wallus@results-hannover.de

February 13, 2003

Table of contents

Chapter 1
Design Guide

Before you start installing or editing OpenCA you should understand the most important principles of OpenCA's design. The first section describes the general design and the second section describes the software design.

This guide will be incomplete at every time because it is the part of documentation which must changed at every time the software changes. So please be lenient toward us :)

1.1 General Design

We start here from scratch to give everybody a chance to understand how OpenCA works. So if you think about these boring guys who write this please take in mind that also OpenCA novices must have a chance to understand the software.

1.1.1 Basic Hierarchy

The basic idea of every X.509 PKI (Public Key Infrastructure) is a strong hierarchical organization. This results a tree of databases if we try to create distributed PKI architecture.

Here we have to add a picture of a databasetree.

Figure 1.1.

The dataexchange between such isolated databases can be handled automatically if you use a distributed databasesystem but in the sense of OpenCA such a distributed databasesystem is only on database in our tree. If you have really isolated database (e.g. for an Offline CA) then you must have a technology for the dataexchange and the management of the complete node in the hierarchy. This management functionalities are bundled in an interface called node or node management. So the design of OpenCA looks like follows

Here we have to add a picture with databases and node management.

Figure 1.2.

Normally every server in the infrastructure of the trustcenter has it's own database for security reasons. This hierarchy is the backbone of the trustcenter.

1.1.2 Interfaces

After you know the basic infrastructure of OpenCA you possibly want to know what we think about such things like CA, RA, LDAP and a public interface which is sometimes called webgateway. OpenCA supports all these softwarecomponents via special web interfaces.

If you want to design a powerfull trustcenter then you must have a concept how you want to organize your workflow. You can see an example in the following picture.

So and here we need a picture with databases and all the interfaces.

Figure 1.3.

OpenCA actually supports the following interfaces:

1. Node (for node management)

2. CA
3. RA
4. LDAP
5. Pub

1.1.2.1 Node

This interface manages the database and handles all the export and import functionalities.

The database can be initialized what means that OpenCA can create all the tables but OpenCA cannot create the database itself because this differs for every vendor. So we need a database with the appropriate accessright and a new database. The interface includes some functions for the backup and recovery of such a node but please bear in mind that you MUST have a seperate backup of the CA's private key and certificate. There is no default mechanism in OpenCA to backup the private key. We don't implement it because first we found no general secure way to backup a private key and second the most CA's use HSMs and therefore you need a completely different and usually proprietary backup strategy.

The export and import will be handled by this interface too. You can configure different rules for the synchronization with nodes on a higher and a lower level of the hierarchy. This includes the configuration of the objects and status which can be exchanged. The configured filters avoid status injections from lower levels of the hierarchy.

1.1.2.2 CA

The CA interface has all function which you need create certificates and CRLs (Certificate Revocation Lists). The CA includes also all functions which you can use to change the configuration via a webinterface. It is not possible to change the configuration via another webinterface.

The CA is the home of the batchprocessors too. OpenCA includes some powerful batchprocessors to create certificates fullautomatically from ERP-systems (Enterprise Ressource Planning - e.g. SAP, HIS, NIS or /etc/passwd).

1.1.2.3 RA

OpenCA's RA is able to handle all kinds of requests. This include things like editing requests, approving requests, creating private keys with smartcards, delete wrong requests and email users.

1.1.2.4 LDAP

The LDAP interface was implemented to seperate the LDAP management completely from the rest of the software. This is necessary because there are many functions which are really special for LDAP admins too because only a few users need these features.

1.1.2.5 Pub

The Public interface include all the small things which the users need. This is only a small list and perhaps it is incomplete

- generates CSRs (certificate signing request) for Microsoft Internet Explorer
- generates CSRs for Mozilla 1.1+ and Netscape Communicator and Navigator
- generates clientindependent requests and private keys (e.g. for KDE's konqueror or server administrators which don't know how to create a private key and request)
- receives PEM-formatted PKCS#10 requests from servers
- enrolls certificates
- enrolls CRLs
- supports two different methods revocation
- search certificates
- tests usercertificates in browsers (Microsoft Internet Explorer and Netscape Communicator and Navigator 4.7x)

1.1.3 Configuration

1.1.3.1 Database

1.1.3.2 Interface

1.1.4 Lifecycle of the objects

Sometimes there are confusions about the status of OpenCA's objects. Therefore you can check here the complete lifecycle of all OpenCA objects. If this figure is not complete, you find a mistake or you don't understand something then please contact openca-users@lists.sf.net.

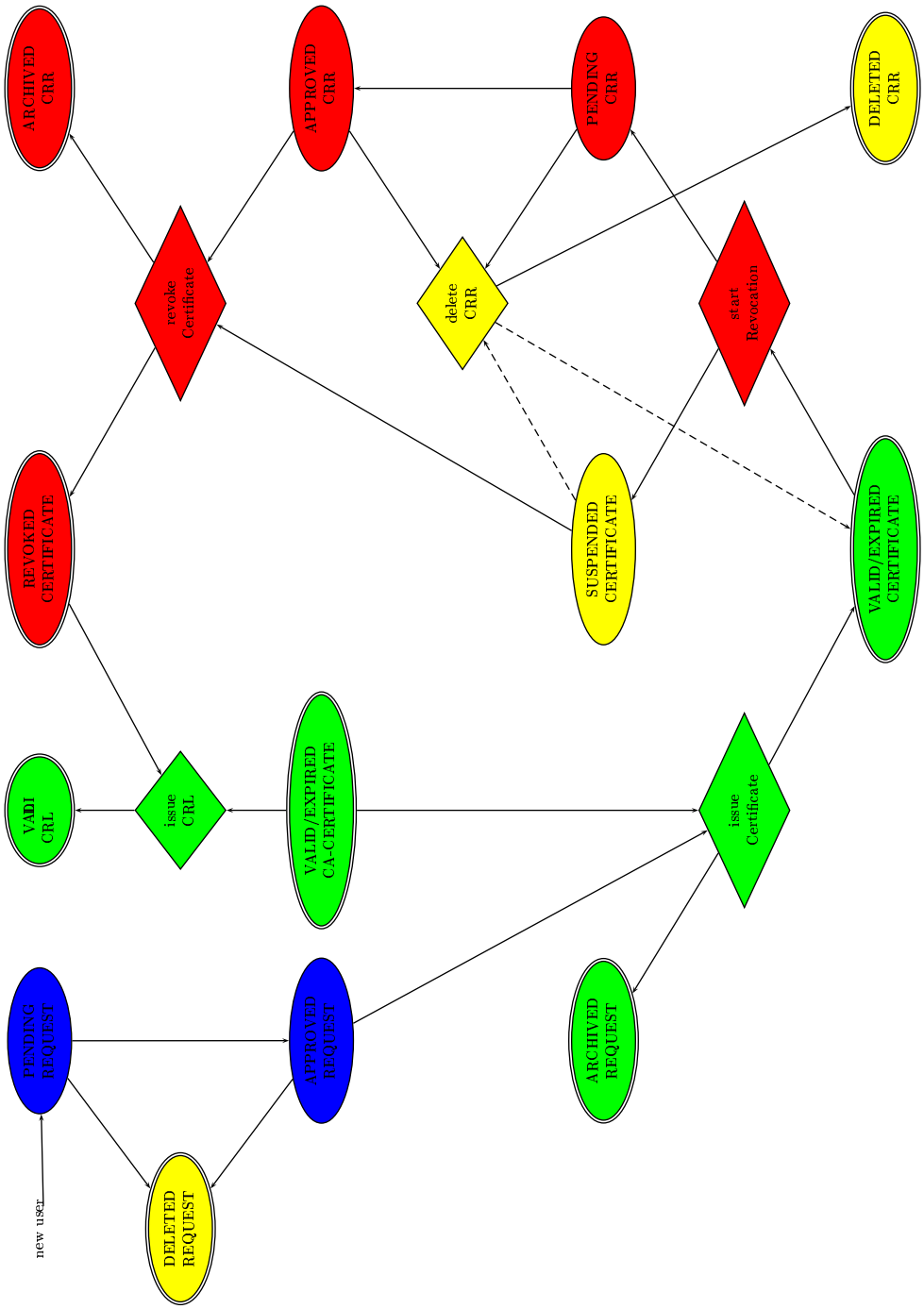

Figure 1.4.

1.1.5 Sub-CA

1.1.5.1 Example 1

Direct chaining of the Root CA and Sub CA including an image.

1.1.5.2 Example 2

Chaining of a Root CA and a Sub CA via a normal public interface.

1.2 Software Design

1.2.1 Database(s)

1.2.2 Interface construction

1.2.2.1 openca.cgi

1.2.2.2 libraries

1.2.2.3 modules

1.2.2.4 commands

1.2.3 Dataexchange and Node management

Chapter 2
Administration Guide

2.1 Installation

The installation of OpenCA is based on autoconf and automake the wellknown GNU-tools. The basic steps are simple.

1. Download a snapshot from OpenCA.
2. Unzip the snapshot (gunzip filename.tar.gz or gzip -u filename.tar.gz).
3. Untar the snapshot (tar -xvf filename.tar).
4. Change into the directory of the snapshot.
5. Configure the sources (see 2.1.2).
6. Run make (please use gmake if you have trouble with your platform specific make).
7. Install the software (see 2.1.3).

The first four steps are quite simple. The problems start with the configuration of the software but before you start reading about the problems please read the requirements carefully because a lot of problems are caused by wrong versions or missing modules.

2.1.1 Requirements

- Perl 5.6.1 or higher
  You can enforce the usage of Perl 5.6.1 during configure if you are using more than one version of Perl on you system. This can happen if the vendor of your operating system ships the system with a very old version of Perl. You must simply call configure in the following way
  env PERL=/path/to/perl ./configure ...
- OpenSSL 0.9.7 20020415 or newer

2.1.2 Configure the sources

There are four big parts in the configuration of the software.

1. placement of the normal OpenCA-software
2. placement of Perlmodules
3. placement of Apache related stuff
4. Preconfiguration of the software

2.1.2.1 Placement of the normal OpenCA-software

Before we start to explain the different options you must understand the general structure. We know a maindirectory which is usually /usr/local/OpenCA. The option to define the directory is —with-openca-prefix. This directory contains several other directories:

- etc

- lib
- var

The names have the meanings like in several other Open Source packages. etc contains the configuration, lib contains the softwareparts which are no binaries and var contains all dynamic data. The binaries (the only exception) are placed by default in /usr/local/bin. You can modify this behaviour by setting –prefix or –exec-prefix. The full structure looks like follows:

```
etc/
etc/openssl/
etc/openssl/openssl/
etc/openssl/extfiles/
etc/rbac/
etc/rbac/modules/
etc/rbac/operations/
etc/rbac/rights/
etc/rbac/roles/
etc/rbac/scripts/
etc/servers/
lib/
lib/cmds/
lib/functions/
lib/servers/
var/
var/crypto/
var/crypto/cacerts/
var/crypto/certs/
var/crypto/chain/
var/crypto/crls/
var/crypto/keys/
var/crypto/reqs/
var/db/
var/mail/
var/mail/crins/
var/tmp/
```

The most important directories for non-developers are etc/servers/ and etc/openssl/.

2.1.2.2 Placement of Perlmodules

The perl modules can be placed in a special directory to remove them easily from your machine after testing OpenCA. The option is –with-module-prefix. We strongly recommend you to set this option to not mix your normal system with OpenCA.

2.1.2.3 Placement of Apache related stuff

We know two different groups of users until today. The users which install OpenCA on a Linux with an Apache with cgi-bin- and htdocs-directory somewhere in the system but with OpenCA on top of the Apache. The other users are the real Apache-admins which compile their Apache by hand and want to embed OpenCA somewhere in the system. We allow the following options to give you an easy but flexible way of installing the software:

General Options

–with-web-host=WEBHOST
: sets the host for all services like www, ldap, ocsp, scep

–with-httpd-host=ARG

sets the hostname of the webserver which hosts the OpenCA-software (if the server is identical with the webhost then you have not to set this option)

–with-httpd-user=ARG
sets the username which uses the Apache. This is needed to set the ownership of the files correctly which must be writable by the software.

–with-httpd-group=ARG
sets the group which uses the Apache

–with-httpd-fs-prefix=HTTPDFSPREIFX
sets httpd filesystem prefix (default is PREFIX/apache)

–with-httpd-url-prefix=HTTPDURLPREFIX
sets httpd URL prefix (default is empty which means OpenCA will be installed in the servers document root)

–with-htdocs-fs-prefix=HTDOCSFSPREFIX
sets htdocs filesystem prefix (default is HTTPDFSPREFIX/htdocs)

–with-htdocs-url-prefix=HTDOCSURLPREFIX
sets htdocs URL prefix (default is HTTPDURLPREFIX)

–with-cgi-fs-prefix=CGIFSPREFIX
sets CGI filesystem prefix (default is HTTPDFSPREFIX/cgi-bin)

–with-cgi-url-prefix=CGIURLPREFIX
sets CGI URL prefix (default is HTTPDURLPREFIX/cgi-bin)

Certification Authority

–with-ca-htdocs-fs-prefix=DIR
sets CA htdocs directory (default is HTDOCSFSPREFIX/ca)

–with-ca-htdocs-url-prefix=DIR
sets CA htdocs URL-prefix (default is HTDOCSURLPREFIX/ca)

–with-ca-cgi-fs-prefix=DIR
sets CA cgi directory (default is CGIFSPREFIX/ca)

–with-ca-cgi-url-prefix=DIR
sets CA cgi URL-prefix (default is CGIURLPREFIX/ca)

Administration of the Online Components

–with-online-htdocs-fs-prefix=DIR
sets Admin htdocs directory (default is HTDOCSFSPREFIX/online)

–with-online-htdocs-url-prefix=DIR
sets Admin htdocs URL-prefix (default is HTDOCSURLPREFIX/online)

–with-online-cgi-fs-prefix=DIR
sets Anline cgi directory (default is CGIFSPREFIX/online)

–with-online-cgi-url-prefix=DIR
sets Admin cgi URL-prefix (default is CGIURLPREFIX/online)

Registration Authority

–with-ra-htdocs-fs-prefix=DIR
sets RA htdocs directory (default is HTDOCSFSPREFIX/ra)

–with-ra-htdocs-url-prefix=DIR
sets RA htdocs URL-prefix (default is HTDOCSURLPREFIX/ra)

–with-ra-cgi-fs-prefix=DIR
sets RA cgi directory (default is CGIFSPREFIX/ra)

–with-ra-cgi-url-prefix=DIR

sets RA cgi URL-prefix (default is CGIURLPREFIX/ra)

LDAP Gateway

–with-ldap-htdocs-fs-prefix=DIR
sets LDAP htdocs directory (default is HTDOCSFSPREFIX/ldap)

–with-ldap-htdocs-url-prefix=DIR
sets LDAP htdocs URL-prefix (default is HTDOCSURLPREFIX/ldpap)

–with-ldap-cgi-fs-prefix=DIR
sets LDAP cgi directory (default is CGIFSPREFIX/ldap)

–with-ldap-cgi-url-prefix=DIR
sets LDAP cgi URL-prefix (default is CGIURLPREFIX/ldap)

Public Gateway

–with-pub-htdocs-fs-prefix=DIR
sets public htdocs directory (default is HTDOCSFSPREFIX/pub)

–with-pub-htdocs-url-prefix=DIR
sets public htdocs URL-prefix (default is HTDOCSURLPREFIX/pub)

–with-pub-cgi-fs-prefix=DIR
sets public cgi directory (default is CGIFSPREFIX/pub)

–with-pub-cgi-url-prefix=DIR
sets public cgi URL-prefix (default is CGIURLPREFIX/pub)

Like you can see it is easy to configure OpenCA for testing only but is also possible to integrate OpenCA into a complex webserver.

2.1.2.4 Preconfiguration of software

All the options which were not described above are used to configure the software itself. This include things like databases, LDAP-servers and names of the installed components (e.g. –with-ra-prefix is the name of the directory with all special softwarecomponents of the Registration Authority in lib/servers/ and the name of the configuration file in etc/servers/).

The details are not only necessary for the software itself. Some parameters are used for the configuration of the certificates. So please be careful if you setup a real trustcenter.

2.1.3 Installing the compiled software

If you want to install the software then you can choose which parts you want to install

- install-ca (only CA and the nodemanagement)
- install-ext (RA, LDAP, Public webinterface and the nodemanagement)
- install-ra (only RA and the nodemanagement)
- install-pub (only Public webinterface and the nodemanagement)

The nodemanagement is always included because you need it to manage the database and the dataexchange with other nodes of the PKI hierarchy.

2.1.4 Example installation of OpenCA

2.1.4.1 Overview

What is what?

- Certificate Authority - CA
  - URL http://ca.intern.results-hannover.de

  - Don't need a connection to any network
- Registration Authority - RA
  - URL https//ra.results-security.de
  - Need connection to the the big badness (internet)
  - contains the public interface for request certificates
  - contains the ra interface for pre task of issue of certs

My browser for everything is netscape 4.79. Perhaps later I test IE 6.0 (capicom.dll).

2.1.4.2 Install the source for ca

OpenCA has two parts: the realy privat one: ca (this one). Here you will really issue certificates. For security reason this ca have not to be in a network. The certificates and the requests must be transported with a floppy. For testing purposes, you can install ca and ra on one server.

Preparations

- If you install SuSE8.1 with standard from Yast you need to install
  - apache
  - make
  - gcc
  - autoconf (optional)
  - mod_perl
  - mod_ssl
  - gettext (optional)
- Installation of openssl on both computers ra and ca
  - → Download openssl version 0.9.7 from www.openssl.org (I use openssl-0.9.7-beta5.tar.gz)
  - → tar xzf openssl-0.9.7-beta5.tar.gz
  - → cd openssl-0.9.7-beta5
  - → ./config
  - → make
  - → make test
  - → make install
  - → for using this new openssl version you must use the path /usr/local/ssl
- Prepare Mysql (I use mysql only on the ra, because then I can easy put the ca into a tar-file!)
  - → look if mysql is running: /etc/init.d/mysql status;
  - → look that mysql starts with the systemstart
    - → yast
    - → system
    - → Runlevel editor
    - → Runlevel properties
    - → set mysql for your default runlevel
  - → mysqladmin password mysqladmin

→ mysqladmin -pmysqladmin create openca

→ mysql -pmysqladmin

→ grant all privileges on openca.* to openca@localhost identified by "mysqlopencapasswd";

→ exit;

- Download openca-software from ftp://ftp.openca.org/pub/openca/v0.9/ on both computers ra and ca
- create dir for the web server files: mkdir /srv/ca
- In the directory perhaps /home/openca
  - tar xzf openca-0.9.1.tar.gz
  - cd openca-0.9.1

Configure the software

WARNING: You have to enter a backslash at the end of every. Until now I don't know how to enter a backslash in TeXmacs :(

```
./configure
     --prefix=/srv/ca
     --with-web-host=ra.results-security.de
     --with-httpd-user=wwwrun
     --with-httpd-group=nogroup
     --with-dist-user=wallus
     --with-dist-group=openca
     --with-ca-organization=security
     --with-ca-locality=Hannover
     --with-ca-country=DE
     --with-service-mail-account=wallus@results-hannover.de
     --with-openssl-prefix=/usr/local/ssl
     --with-mailprogram="/usr/sbin/sendmail -t"
     --with-hierarchy-level=ra
     --enable-dbi
     --with-db-type=mysql
     --with-db-name=openca
     --with-db-host=localhost
     --with-db-port=3306
     --with-db-user=openca
     --with-db-passwd=mysqlopencapasswd
```

Compile and Install CA

- make
- make install-ca

If you want make everything again, don't forget to delete the installed files: rm -rf O* apache bin man

Configuration of apache CA Server

I use virtuell hosting from httpd.conf.

```
...
BindAddress 192.168.251.202:80
Listen 192.168.251.202:80
...
include /srv/ca/apache.conf
```

```
....
# Clear other part of virtuell hosts, if you don't use ist.
#ca.intern.results-hannover.de must be in DNS.
#From /srv/ca/apache.conf

<VirtualHost ca.intern.results-hannover.de:80>
    ServerAdmin wallus@results-hannover.de
    DocumentRoot /srv/ca/apache/htdocs
    ServerName ca.intern.results-hannover.de
    SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown downgrade-
1.0 force-response-1.0
    CustomLog /var/log/httpd/ssl_request_log "%t %h %{SSL_PROTOCOL}x
%{SSL_CIPHER}x "%r" %b"
    <Directory "/home/openca/apache/htdocs">
        Options Indexes FollowSymlinks MultiViews
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>
    ScriptAlias /cgi-bin/ "/srv/ca/apache/cgi-bin/"
    <Directory "/srv/ca/apache/cgi-bin">
        AllowOverride None
        Options None
        Order allow,deny
        Allow from all
    </Directory>
</VirtualHost>
```

chown wwwrun:nogroup /srv/ca/apache.conf

2.1.4.3 Install the source for ra

The ext interface: Here you can create an certrequest (look at /pub) and sign a request (as ra-operator, at /ra). After this the request must be transported to the ca.

Preparations

Please see Preparations at 2.1.4.2

Configure the software

WARNING: You have to enter a backslash at the end of every. Until now I don't know how to enter a backslash in TeXmacs :(

```
./configure
     --prefix=/srv/ca
     --with-web-host=ra.results-security.de
     --with-httpd-user=wwwrun
     --with-httpd-group=nogroup
     --with-dist-user=wallus
     --with-dist-group=openca
     --with-ca-organization=security
     --with-ca-locality=Hannover
     --with-ca-country=DE
     --with-service-mail-account=wallus@results-hannover.de
     --with-openssl-prefix=/usr/local/ssl
     --with-mailprogram="/usr/sbin/sendmail -t"
     --with-hierarchy-level=ca
     --enable-dbi
```

```
        --with-db-type=mysql
        --with-db-name=openca
        --with-db-host=localhost
        --with-db-port=3306
        --with-db-user=openca
        --with-db-passwd=mysqlopencapasswd
        --with-ldap-host=ra.results-security.de
        --with-ldap-root="cn=Manager,ou=security,o=results-hannover,c=de"
        --with-ldap-root-pwd=ldappasswd
```

Compile and Install RA

- make
- make install-ext

Configuration of apache RA Server

```
   #I use virtuell hosting:
   #From httpd.conf:
   ....
   BindAddress 62.48.68.157:443
   Listen 62.48.68.157:443
   .....
   include /srv/ra/apache.conf
   ....
   # Clear other part of virtuell hosts, if you don't use ist.
   #ra.results-security.de must be in DNS.

   #From /srv/ra/apache.conf
   <VirtualHost ra.results-security.de:443>
       ServerAdmin wallus@results-hannover.de
       DocumentRoot /srv/ra/apache/htdocs
       ServerName ra.results-security.de
       SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown downgrade-
1.0 force-response-1.0
       SSLEngine on
       SSLCertificateFile /srv/ra/ssl.crt/server.pem
       SSLCertificateKeyFile /srv/ra/ssl.key/key.pem
       <Directory "/srv/ra/apache/htdocs/pub/">
           Options Indexes FollowSymlinks MultiViews
           AllowOverride None
           Order allow,deny
           Allow from all
       </Directory>

       <Directory "/srv/ra/apache/htdocs/ra/">
           Options Indexes FollowSymlinks MultiViews
           AllowOverride None
           Order allow,deny
           Allow from 62.48.68
       </Directory>
       <Directory "/srv/ra/apache/htdocs/ra_node/">
           Options Indexes FollowSymlinks MultiViews
           AllowOverride None
           Order allow,deny
           Allow from 62.48.68
       </Directory>
       <Directory "/srv/ra/apache/htdocs/ldap/">
```

```
        Options Indexes FollowSymlinks MultiViews
        AllowOverride None
        Order allow,deny
        Allow from 62.48.68
    </Directory>
    ScriptAlias /cgi-bin/ "/srv/ra/apache/cgi-bin/"
    <Directory "/srv/ra/apache/cgi-bin">
        AllowOverride None
        Options None
        Order allow,deny
        Allow from all
    </Directory>
</VirtualHost>
```

Note: The ra and ldap interface schould be only accessible from ra. I use here simple method for protecting; Allow from <my subnet>.

This is not secure enough and must not be used in a real production enviroment. Better way is to give access only the clients with the appropiate ra operator cert. But the cgi script need also some protect rules.

Configuration of LDAP server

On ra computer from /etc/openldap/slapd.conf:

```
include /etc/openldap/schema/core.schema
include /etc/openldap/schema/cosine.schema
include /etc/openldap/schema/inetorgperson.schema

pidfile  /var/run/slapd/slapd.pid
argsfile /var/run/slapd/slapd.args

allow    bind_v2
database bdb
suffix   "ou=security,o=results-hannover,c=de"
rootdn   "cn=Manager,ou=security,o=results-hannover,c=de"
rootpw   ldappasswd
```

This must be in accord with the lines in /srv/ra/OpenCA/etc/servers/ldap.conf:

```
## Now the LDAP default base dn
basedn "ou=security,o=results-hannover,c=de"

## Let's define the privileged Account Allowed to Modify the LDAP entries
ldaproot "cn=Manager,ou=security,o=results-hannover,c=de"
ldappwd "ldappasswd"
```

Start your slapd /etc/init.d/ldap start with yast / system / Runlevel editor / Runlevel properties / set ldap for you default runlevel

Note: I have still problems with ldap!

2.1.4.4 Initialize ca

Using netscape 4.79.

Preparations

On SuSE 8.1 or if you run an X-Server the default-permissions on the floppy device are not correct. Do this, if you get problems to write on floppy (also when you restart your X-Server):

chmod wwwrun /dev/fd0

Intialize CA Phase 1

→ Initialize the Certification Authority on http://ca.intern.results-hannover.de

→ Initialize Database

→ Create new key -> des3,2048, ca-passwd

→ Generate new CA Certificate Request (use generated secret key)

→ email:wallus@results-hannover.de

→ Common name: ca

→ organizational unit: intern

→ Organization: results-hannover.de

→ Country: DE

Resulting DN= "emailAddress=wallus@results-hannover.de, CN=ca, OU=intern, O=results-hannover.de, C=DE"

→ Generate Self Signed CA Certificate (from altready generated request) -> 730, ca-passwd

→ Rebuild CA Chain

→ Put in a floppy disk, make clear, that wwwrun has right to write on /dev/fd0 (chown wwwrun /dev/fd0)

→ Export Configuration

Intialize CA Phase 2

→ Initialize/Create the initial administrator (user-cert for signing transactions) on http://ca.intern.results-hannover.de

→ Create a new request

→ Email: wallus@results-hannover.de

→ Name: Harald Wallus

→ Certicate Request Group: Internet

→ Role: RA Operator !Change from default

→ Registration Authority: Trustcenter itself

→ PIN: download-passwd

→ Choose a keysize: 1024 ! Change from default

→ 2 x Continue

→ Edit the request

→ OK

→ issue the certificate, ca-passwd

→ Handle the certificate

→ Certificate and Keypair: change to PKCS#12

→ Download, download-passwd store it in ra_datum.p12 (this is for netscape4.79)

→ Load the certificate into the browser

→ Netscape menu

→ Communicator

- → Tools
- → Security-Info
- → Yours
- → Certificate import
  - → Password of your cert storage
  - → Choose the file above
  - → Download passwd
  - → short name for your brain like raop_datum
  - → OK
  - → close

Intialize CA Phase 3

- → Initialize/Create the initial RA certificate (web cert for apache) on http://ca.intern.results-hannover.de
- → Create a new request
  - → Email: wallus@results-hannover.de (email adress of webmaster)
  - → Name: ra.results-security.de (name of the webserver)
  - → Certicate Request Group: Internet
  - → Role: Web Server ! Change from default
  - → Registration Authority: Trustcenter itself
  - → PIN: download-passwd
  - → Choose a keysize: 1024 ! Change from default
  - → 2 x Continue
- → Edit the request
  - → change Subject alternate Name from "email:wallus@results-hannover.de" to "DNS:ra.results-security.de;email:wallus@results-hannover.de" (DNS must be written in upper letters because OpenSSL is case sensitive)
  - → OK
  - → issue the certificate, ca-passwd
- → Handle the certificate
  - → Certificate and Keypair: SSLeay (mod_ssl), download , user-passwd
  - → The Cert appears in the browser
  - → Copy it with your mouse
  - → store the first, public part (BEGIN CERTIFICATE till END CERTIFICATE) into /home/ra/ssl.cert/server.pem and the privat part (BEGIN RSA.... till END RSA ....) into /home/ra/ssl.cert/key.pem
    - → mkdir /home/ra/ssl.crt
    - → vi /home/ra/ssl.crt/server.pem
    - → mkdir /home/ra/ssl.key
    - → vi /home/ra/ssl.key/key.pem

2.1.4.5 Initialize ra

Using netscape 4.79.

Preparations

- Prepare Mysql (I use mysql only on the ra, because then I can easy put the ca into a tar-file!)
  - → look if mysql is running: /etc/init.d/mysql status;
  - → look that mysql starts with the systemstart
    - → yast
    - → system
    - → Runlevel editor
    - → Runlevel properties
    - → set mysql for your default runlevel
  - → mysqladmin password mysqladmin
  - → mysqladmin -pmysqladmin create opencara
  - → mysql -pmysqladmin
    - → grant all privileges on opencara.* to openca@localhost identified by "mysqlopencapasswd";
    - → exit;
- Be sure, that your ra computer can send emails.

Export Configuration from CA

- → Put in a freshly formatted floppy
- → Make shure that wwwrun has access to /dev/fd0 (chown wwwrun /dev/fd0)
- → Open within the browser http://ca.intern.results-hannover.de/ca_node/
- → Dataexchange
- → Enroll data to a lower level of the hierarchy
- → Configuration

Initialize Database of RA

- → Point your browser to https://ra.results-security.de/ra_node/
  - → Server init
  - → Initialize Database

  Hint: If you need to initialize the database once again, you must recreate the database.
- → Now you have to put the floppy from ca, created above, in to the ra computer
- → configure your ldap-Server (look at the txt file ldap configuration)
- → make shure that wwwrun has access to /dev/fd0 (chown wwwrun /dev/fd0)
- → Import the configuration
  - → Server init
  - → Import Configuration

If you must perform the actions again then you should read the following hints:

- → shut down slapd
- → remove all files beneath /var/lib/ldap
- → start slapd
- → drop database opencara
- → create the old database new

→ the permits still exists in the db

→ initialize the database of ra

→ import the configuration from floppy

2.1.4.6 Create a Cert for a Webserver

Preparations

- Please Control if your have set on all three flags on your cert
  Example for Netscape 4.79:
  → Communicator
  → Tools
  → Security info
  → Certificates/Signer
  beneath: This Certificate belongs to a Certifying Authority
  → You must set on all three flags
- Control if you have importet you ra operator cert
  Example for Netscape 4.79:
  → Communicator
  → Tools
  → Security info
  → Yours

Create a basic request

→ Point your browser to https://ra.results-security.de/pub/
Hint: Because we have imported the ca cert before we don't need that now.

→ Request a Certificate

→ Basic Request (wallus@results-hannover.de, rubixml.intern.results-hannover.de, Internet, Webserver, Trustcenter itsself, pin, pin , 1024)

→ 2 x continue

Approve this request by RA

→ Point your browser to https://ra.results-security.de/ra/

→ Certificate Requests, Trustcenter itself

→ continue

→ You see one certificate Request.

→ Click on serial number

→ Edit request
change the Subject alternative name from "email:wallus@results-hannover.de" to "DNS:rubixml.intern.results-hannover.de;email:wallus@results-hannover.de"

→ OK

→ Approve and Sign Request

Export the request from RA to CA

→ Put in a fresh formatted floppy into ra.

→ Point your browser to https://ra.results-security.de/ra_node/ (Server Management)

→ Dataexchange

→ Upload data to a higher level of the hierarchy

→ Requests

Your approved request is written on to floppy.

Preparations on the CA to create a webserver certificate

Make sure that you have importetd the certificate from the CA and that you trust it completly (all three flags in netscape).

Import the request into ca

→ Put the the floppy into the ca computer.

→ Point your browser to http://ca.intern.results-hannover.de/ca_node/ (Server Management)

→ Dataexchange

→ Receive data from a lower level of the hierarchy

→ Requests

Issue the certificate

→ Point your browser to http://ca.intern.results-hannover.de/ca/

→ Approved Certificate Requests

→ Click onto the serial number (not onto Op.)

→ Issue the cert (Input CA password)

If you get an error, make shure you don't misspelled the password. You can check Apache's error_log to get more informations.

→ Check the new certificate

→ Certificates

→ Valid Certificates

→ Click on the appropiate serial number.

→ Here you can download the cert too but we will use the usual way via the RA.

Export the cert from CA and import it to RA

→ Point your browser to http://ca.intern.results-hannover.de/ca_node/ (Server Management)

→ Dataexchange

→ Enroll data to a lower level of the hierarchy

→ Certs

→ Now we put this floppy into ra computer.

→ Point your browser to https://ra.results-security.de/ra_node/ (Server Management)

→ Dataexchange

→ Download data from a higher level of the hierarchy

→ Certs

→ Email new users

E-mail new users (make shure, that your ra can send mails)

Download the cert

It is not possible to download a web server cert via the pub interface if you generated the key on the server (basic request does this by default).

→ Point your browser to https://ra.results-security.de/ra/

→ Certificates

→ Valid Certificates

→ Choose the appropiate serial

→ put the cert with copy/paste into the appropiate webserverfiles.

If you send a PKCS#10 request via the public interface to OpenCA the you can do the following to download the certificate:

→ https://ra.results-security.de/pub/

→ Search the certificate (enter the appropriate data)

→ click on the correct serial

→ shift + Download (the shift is important because some browsers like Netscape detect the MIME-type)

2.2 Configuration

This chapter describes how you can configure OpenCA.

2.2.1 OpenSSL

You must care about three configurationfiles and -directories etc/openssl/openssl.cnf, etc/openssl/openssl/ and etc/openssl/extfiles/. The first file contains the configuration for the CA. This means the file is used for the generation of the initial CA-CSR, the selfsigned certificate (if you setup a Root-CA) and the CRLs. The file is configured fullautomatically during the installation but if you are setting up a serious CA then you should check this file too. The directory etc/openssl/openssl/ contains the configuration for the different roles except of the extensions. The relevant things which you must compare with your policy are the lifetime of the certificate and the algorithm which is used to sign the certs. The dircetory etc/openssl/extfiles/ contains the definitions of the extension. Please check these files carefully.

2.2.1.1 Servercertificates

The different names of HTTPS-Servers are one of the most problematic things in the todays world. Like for many other cryptographic issues in the web there is a standard for servercertificates - RFC 2818 "`HTTP over TLS`".

The standard defines that you have to check the subject alternative name for an appropriate entry (DNS or IP see RFC 2459). If this search fails then check the common name in the distinguished name of the certificate.

If you use Microsofts Internet Explorer then you have no problems. The IE is full standard compliant. The problem is Netscape. They use the common name like a regular expression in Unix. The common name can be in the format (server1|server2).my_domain.org. The clever ones would argue now that we must simply set the subject alternative name like defined by RFC 2818 and set a normal common name because the subject alternative is checked first. This is a nice idea but Netscape ignores the subject alternative name if it checks the name of the server versus the content of the certificate.

The solution is a mix between RFC 2818 and Netscape behaviour. You must set the common name in the distinguished name like Netscape defines it. After this you must set all DNS-names and the IPs of the server in the subject alternative name. If you do this then all standard compliant browsers will evaluate the subject alternative name first and will ignore the common name in the distinguished name. So the certificate is standard compliant but supports the cruel behaviour of Netscape too.

2.2.1.2 F-Secure VPN+

If you want to use OpenCA with F-Secure VPN+ then you should bear in mind that this software can only download a CRL via http or ldap. They don't support https and ldaps. This is important if you configure your CRLDistributionPoints in OPENCADIR/etc/openssl/extfiles/*.ext. You can easily fix this problem by using LDAP for CRL-distribution.

2.2.1.3 Troubleshooting

All cryptographic opertions fails because the software cannot find openssl.
Please check the configured path to the binary of OpenSSL in etc/servers/*.conf. The variable which is the problem is OPENSSL.

Apache's error.log reports a nonexistent option "-subj" of openssl req.
You must install and use OpenSSL v0.9.7 or later. This is necessary because some options are only supported in the later versions. OpenSSL v0.9.6 or earlier contains some bugs in the definitions of the objects. This can result in displaying wrong DNs. The certs are correct! The last major change in the definitions was at March the 28th in 2002.

2.2.2 Database

If you use OpenCA::DB (the default) then you can ignore this. If you use a SQL-database then you should read this perhaps. You can configure your database directly via etc/servers/DBI.conf or you set the options during the installation. You must not set all options for all databases. Some databases locate things like the host or the port automatically by their own configuration-files (e.g. Oracle or DB2).

2.2.2.1 Troubleshooting

Internal Server Error.
The apache reports an internal servererror just for the initial CA-page. This happens if the databasemanagementsystem is not running. Please start the database :)

The document contains no data.
This can happen if you forget to install a perlmodule or a databasedriver itself or there is misconfiguration. You need at minimum the following:

- DBI
- DBD::DB_Vendor
- a driver of the databasevendor must also be installed normally

Apache's error.log contains a message from IBM DB2 that the environment is not setted.
Please check the settings in etc/servers/DBI.conf because this happens if IBM's software cannot find the libraries and databases.

2.2.3 Distinguished Name

The following options influence the DN:

- OrganizationUnit (1.OU, 2.OU ...)
- Organization
- Country
- Locality
- SET_REQUEST_SERIAL_IN_DN and REQUEST_SERIAL_NAME enforce the inclusion of the request's serial in the DN
- SET_CERTIFICATE_SERIAL_IN_DN and CERTIFICATE_SERIAL_NAME enforce the inclusion of the certificate's serial number in the DN (strongly recommended)
- DN_WITHOUT_EMAIL (e.g. S/MIME v3)
- the basic CSR will be described extra because this influence only the requestgeneration and not the certificategeneration directly

Annotation:
OpenCA use internally the DNs like described in RFC 2253. This is the opposite order like OpenSSL uses. The transformation from RFC 2253 to the order which OpenSSL uses is handled fullautomatically.

2.2.3.1 dc-Style

OpenCA uses by default the old "o=University,c=de"-Style. Several users like international companies, universities or other big organizations need the new dc-style. Therefore we support the dc-style too. It is necessary to change several files because the configuration of the DNs is highly integrated into the software. We will explain it with an example.

```
base dn or suffix: dc=university,dc=edu
user dn: dc=mike tester,dc=university,dc=edu
webserver dn:dc=www,dc=university,dc=edu
ca dn:dc=CA,dc=university,dc=edu
```

There are five things which you have to check for the change to the dc-style. The steps will be now described:

1. etc/servers/*.conf

   There are two things which must be changed in the configuration files of the servers.

   The LDAP configuration must be adapted to the new dc-style. The variables - which you must modify - are basedn and ldaproot. The basedn is the suffix of the LDAP server. The ldaproot is the dn of the user root to bind to the LDAP server. The ldaproot has not to be changed because it is freely configurable by the administrator of the LDAP server.

   ```
   basedn "dc=university,dc=edu"
   ldaproot "dc=manager,dc=univesity,dc=edu"
   ```

   The configuration of the requests must be changed too because they are prepared for the old style. Please read the following example to get an overview of a dc-styled configuration. Please read the section 2.2.6 to understand how the normal requests can be configured.

   ```
   DN_TYPE_BASIC_BODY "YES"
   DN_TYPE_BASIC_KEYGEN_MODE "SERVER"
   DN_TYPE_BASIC_KEYGEN_SHEET
"/usr/local/OpenCA/lib/servers/pub/sheets/basic_csr_confirm_request.html"

   DN_TYPE_BASIC_BASE "DC" "DC"
   DN_TYPE_BASIC_ELEMENTS "DC"

   DN_TYPE_BASIC_NAME "Basic User Request"

   DN_TYPE_BASIC_BASE_1 "University"
   DN_TYPE_BASIC_BASE_2 "edu"

   DN_TYPE_BASIC_ELEMENT_1 "Name"
   ```

2. main.html

   Please check the installed or prepared files with the name main.html because several HTML files display the suffix of all the DNs.

3. certsMail.txt

   You can find this file in lib/servers/ra/mails/. It is the default template for the mails which RA Operators can send to the users. It includes the suffix of the LDAP server. This suffix is called Dir Root. This suffix must be changed according to real suffix of your LDAP server.

4. OpenSSL configuration

   You must modify the files OPENCADIR/etc/openssl/openssl.cnf and OPENCADIR/etc/openssl/openssl/*.conf. The policy and req sections must be changed to support requests and certificates with DNs in the dc-style. If you don't know how to configure OpenSSL then please read the documentation of OpenSSL.

5. CA CSR (certificate signing request)
   If you generate the initial request for the CA request then please ignore all the field for the normal DN-style. Simply enter nothing in all field until the software displays the window which show you the complete DN. There you have to enter the complete DN of the CA request. The DN is in RFC 2259 format and all DC must be written in big letters because OpenSSL is case sensitive.

2.2.4 Subject Alternative Name

The most people want to use OpenCA for issuing certificates to users or they want to test a PKI without buying a commercial trustcenter only for testing. So they want create a request, approve the request and issue a certificate. The problem is that they forget to the edit the request and the subjct alternative name was not set.

OpenCA knows two switches in ca.conf to set the subject alternative name automatically. The switch AUTOMATIC_SUBJECT_ALT_NAME enables the mechanism to set the subject alternative name automatically if it was not set in the header of the request. The second switch DEFAULT_SUBJECT_ALT_NAME defines the type of the default value. Actually we implemented only support for the emailaddress. If you need support for DNS name(s) or IP addresse(s) then contact us. We only don't implement it because nobody need it until now.

2.2.5 LDAP

OpenCA provides an LDAP interface for users to download certificates from a central repository. This interface can be utilised by browser address books and specialised LDAP clients.

Before the OpenCA RA components can write certificates and CRLs to the directory you must have an LDAP compliant directory installed and available to the RA components (this can be on the same or different machine). One example of an appropriate directory is the OpenLDAP project.

2.2.5.1 Configuration of the Directory

A full description of the configuration of your LDAP directory is outside the scope of this document. Important points to note are:

- Ensure that the following schemas are included (probably in the slapd.conf file):
  - core.schema
  - cosine.schema
  - inetorgperson.schema
- Ensure the directory is started with the appropriate suffix (e.g. o=myorg,c=gb).
- Ensure the rootdn is specified.
- Ensure the root password is specified.

2.2.5.2 Configuration of the RA components

Two configuration files must be configured for the RA component to make use of the LDAP directory to store certificates; OPENCADIR/etc/servers/online.conf and OPENCADIR/etc/servers/ldap.conf. OpenCA use the following variables to configure the LDAP-server (ensure the following entries match your ldap server):

LDAP
If this option is set to ON or YES then the LDAP-code of OpenCA will be activated.

updateLDAPautomatic
If this option is different from OFF and NO then OpenCA updates the LDAP during the import of data from the CA.

ldapserver

This option contains the host where your LDAP-server runs.

ldapport
This option contains the port where your LDAP-server listens (usually 389).

basedn
This option contains the base-DN of your LDAP-server. It is the root of your LDAP-server. This is the same thing which is called suffix in OpenLDAP's configurationfile.

ldaproot
This is the DN of the user which OpenCA uses to bind to the LDAP-server and add or remove entries. The most user set here the rootuser of the LDAP-server but this is not mandatory.

ldappwd
This is the passphrase for the DN which is used to bind to the server (ldaproot). Actually this is a cleartext passphrase.

2.2.5.3 Writing Certifciates to the Directory

As long as the "updateLDAPautomatic" entry is set to "yes" the RA will attempt to upload certificates to the directory after an import. Before this can happen the directory must be initialised and the appropriate structure must be implemented. In this version of OpenCA this initialization is done automatically.

2.2.5.4 Troubleshooting

Connection refused.

This occurs if OpenCA cannot make a connection to the LDAP directory. Make sure that the ldap server is running and is listening on the correct port. Make sure that the settings in ldap.conf and online.conf match your ldap server settings.

Bind failed. Errorcode 49.

A connection has been made to the ldap server, but the credentials to log into the server as admin are wrong. The bind operation is performed after the connection. Check the ldaproot (the LDAP administrator's DN) and ldappwd (the password of the ldap administrator).

The resultcode of the nodeinsertion was 65.

This sometimes means that the RA could not insert the appropriate entry for a certifciate (the exact definition is LDAP_OBJECT_CLASS_VIOLATION). Check that you have the directory started with the appropriate schemas (core, cosine and inetorperson) this is usually found in the slapd.conf file.

More debugging information can be seen by turning on debugging in ../lib/finctions/ldap-utils.lib (i.e. DEBUG = 1;).

2.2.5.5 Sourcecodeorganization

Structure of the code

```
Scripts for the import |
   of certificates     |
                       |
-----------------------| scripts to add
                       | objects to LDAP
   export-import.lib   |
                       |
```

```
---------------------------------------

                ldap-utils.lib
```

The relevant commands
addCertsLDAP (puts all valid certs to LDAP)
addCrlLDAP (puts all CRLs to LDAP)
importAllFromCA (via export-import.lib)
importCRL (via export-import.lib)
importCerts (via export-import.lib)
importCertsLDAP (puts all certs from the last import to LDAP)
importConfig (puts CA-certs to LDAP)
updateCACertsLDAP (update the CA-certificates on the ldap server)
updateCRLonLDAP (writes the most actual CRL to LDAP)
updateCertsLDAP (writes/removes the user-certificates to/from LDAP)
updateLDAP (puts all certs from the last import to LDAP)
(oh, we have a redundancy here updateLDAP and addCertsLDAP do the same) (updateLDAP is reserved for the future so set all links etc. to importCertsLDAP) (addCertsUser should not be a function of ldap-utils.lib)

export-import.lib
eximObjectToLDAP

ldap-utils.lib
addCertsUsers (will be moved to importCertsLDAP)
addLDAPobject (takes a cert and create the necessary nodes in the LDAP)
addLDAPattribute (add certs and CRLs to the LDAP)
deleteLDAPattribute (remove certificates from LDAP)

2.2.6 Configuration of CSRs

The basic CSR is the most flexible way in OpenCA to create CSRs. Let's start with an example:

```
Basic_CSR_Keysizes "512" "768" "1024" "2048" "4096"

DN_TYPES "BASIC"
DN_TYPE_BASIC_KEYGEN_MODE "SERVER"
DN_TYPE_BASIC_KEYGEN_SHEET "@lib_prefix@/servers/@pub_prefix@/sheets/basic_csr_c
onfirm_request.html"

DN_TYPE_BASIC_BODY "Y"
DN_TYPE_BASIC_BASE "O" "C"
DN_TYPE_BASIC_ELEMENTS "emailAddress" "CN" "OU"
DN_TYPE_BASIC_NAME "Basic User Request"
DN_TYPE_BASIC_BASE_1 "@ca_organization@"
DN_TYPE_BASIC_BASE_2 "@ca_country@"
DN_TYPE_BASIC_ELEMENT_1 "E-Mail"
DN_TYPE_BASIC_ELEMENT_2 "Name"
DN_TYPE_BASIC_ELEMENT_3 "Certificate Request Group"
DN_TYPE_BASIC_ELEMENT_3_SELECT "Internet" "Partners" "Employees" "Trustcenter"
```

The first line defines the available keysize. The next variable DN_TYPES defines the available configurations of basic_csr. The command basic_csr is called via a link and the link must contain an option CSR_TYPE which defines the configuration which is used for this CSR. If you don't set this option then basic_csr starts it's browserdetection.

The default type which is supported by OpenCA is BASIC. You can simply add a type and set a correct link in the public gateway. You can find an example on the public gateway by looking at the link Basic Request.

The prefix of every definition is now DN_TYPE_BASIC. The NAME defines the displayed name (e.g. "Request for managers only"). The BODY defines the type of the request. If the value is Y or YES then a key and a request will be stored and if necessary generated. If the value is N or NO then only a header will be generated. This option is used to get the necessary data from a user to initialize a smartcard on the registration authority.

DN_TYPE_BASIC_KEYGEN_MODE specifies the way how to generate a key and request. The supported modes are SERVER, SPKAC and IE. SPKAC is used with Mozilla and Netscape, IE is used for Microsoft Internet Explorer and SERVER is used for all other situations.

The BASE is the part of the DN which is not editable by the user who requests a certificate. The other BASE_numbers define the values of the elements which are used for the not editable part of the DN.

The ELEMENTS is the part of DN which can be defined by the user. The ELEMENT_numbers are the displayed names of the elements. The normal user don't know what is a CN or a commonName. The most users will be confused if they see two fields with the same name (e.g. OU). All fields are textfields by default. You can specify ELEMENT_number_SELECT followed by a list of values. OpenCA creates a HTML-select from this definition.

2.2.7 Dataexchange

The dataexchange of OpenCA is highly configurable. So first we have to describe some general concepts.

If you look at OpenCA from a database viewpoint then OpenCA is a tree of hierarchical organized databases. Every database is used by some web interfaces. So one node of the hierarchy consists of a database and some web interfaces. If we describe the dataexchange then we describe the dataexchange between nodes. This is the reason why we called the management interface node.

A node can exchange data with a node of a higher level of the hierarchy or with several nodes which are on a lower level of the hierarchy. If export data to a higher level of the hierarchy then we UPLOAD data and if we import data from such a node then we DOWNLOAD data. If import data from a lower level then we RECEIVE data and if export data to such a node then we ENROLL data.

If you exchange object in a security relevant area then you must define which object with which state you want to exchange. Therefore you can define in OpenCA which objects with which state you accept from which direction. Also OpenCA allows only to overwrite existing objects if you DOWNLOAD CA-certificates, CRLs, CSRs and CRRs. Status or object injections are not accepted in all other situations. OpenCA includes some default configurations to help you on the way to secure configuration.

The following example is for the import from a higher level of the hierarchy:

```
DOWNLOAD_CA_CERTIFICATE_STATES VALID
DOWNLOAD_CERTIFICATE_STATES    VALID
DOWNLOAD_CRL_STATES            VALID
DOWNLOAD_CRR_STATES            ARCHIVED DELETED APPROVED
DOWNLOAD_CSR_STATES            ARCHIVED DELETED
DOWNLOAD_MAIL_STATES           CRINS DEFAULT
```

The export/import technology itself is another important aspect. You can configure OpenCA to use different methods for the dataexchange with higher and lower levels of the hierarchy. Therefore we implemented options to prepare and cleanup IO operations. This is necessary if you have to start special networkinterfaces or to mount devices. There are also options to configure the EXPORT and IMPORT of the directory with the data and there is an option to TEST the result. EXPORT, IMPORT, START and STOP accept more then one argument. Every argument will be seperately executed. The parameters @__SRC__@ and @__DEST__@ include the directories with the data. The parameter @__DEVICE__@ will be replaced with the value of DEVICE.

A sample configuration for the dataexchange with a higher level could be:

```
EXPORT_IMPORT_UP_DEVICE "/dev/fd0"
EXPORT_IMPORT_UP_START ""
EXPORT_IMPORT_UP_STOP ""
EXPORT_IMPORT_UP_EXPORT "/bin/tar -cvfp @__DEVICE__@ -C @__SRC__@"
EXPORT_IMPORT_UP_IMPORT "/bin/tar -xvf @__DEVICE__@ -C @__DEST__@"
EXPORT_IMPORT_UP_TEST "/bin/tar -tvf @__DEVICE__@"
```

The hierarchylevel LOCAL is used for backups, batchprocessors and such things. It is also possible to configure OpenCA for the use with temporary private networks. Here is another example for a CA:

```
EXPORT_IMPORT_DOWN_DEVICE "openca.tar"
EXPORT_IMPORT_DOWN_START "/sbin/ifconfig eth0 up"
EXPORT_IMPORT_DOWN_STOP "/sbin/ifconfig eth0 down"
EXPORT_IMPORT_DOWN_EXPORT "/bin/tar -cvfp /usr/local/openca/var/tmp/@__DEVICE__@
-C @__SRC__@" "/usr/bin/scp /usr/local/openca/var/tmp/@__DEVICE__@
openca@ra.openca.org:/usr/local/OpenCA/var/tmp/" "rm
/usr/local/openca/var/tmp/@__DEVICE__@"
EXPORT_IMPORT_DOWN_IMPORT "/usr/bin/scp
openca@ra.openca.org:/usr/local/OpenCA/var/tmp/@__DEVICE__@
/usr/local/openca/var/tmp/@__DEVICE__@" "/bin/tar -xvf
/usr/local/openca/var/tmp/@__DEVICE__@ -C @__DEST__@" "rm
/usr/local/openca/var/tmp/@__DEVICE__@"
EXPORT_IMPORT_DOWN_TEST ""
```

2.2.8 RBAC

Please read the whole chapter about RBAC.

2.2.9 Module-ID

Every module in OpenCA has a module-ID. This ID you can find in the configurationfile of a module. The ID is used to create unique serial numbers for the requests. The moduleshift defines how many bits at the beginning of a serial number (the least significant bits) are reserved for the module's ID. The advantage is that you can issue a request at any module of OpenCA without synchronizing the databases at every time you issue a request because the module's Id is part of every request serial. The parameter in the configurationfiles are ModuleID and ModuleShift. You can configure both parameters via ./configure. The options are –with-module-shift, –with-ra-module-id and –with-pub-module-id. The ID of the CA is at every time zero.

2.3 RBAC

The RBAC-code is one of the most difficult parts of OpenCA. You must understand at minimum the logic before you setup the software.

2.3.1 Setup

This is a short step-by-step guide. Don't use it without understanding the logic of the software.

1. Initialize the certification authority
2. Create the initial administrator
3. Create the initial RA certificate

4. Export the configuration from the CA
5. Import the configuration into the RA
6. Activate the verification of the client in the Apache's configurationfile (SSLVerifyClient)
   Please take in mind that your Apache must set the environmentvariable SSL_CLIENT_CERT which you can activate by adding +ExportCertData to SSLOptions.
7. Activate the RBAC-mechanism of OpenCA in OPENCADIR/etc/servers/online.conf and OPENCADIR/etc/servers/ra.conf

Actually the RBAC is preconfigured for use with the web-interfaces Online and RA. Both interfaces are configured to be the RBAC-module "RA 1". So they are logically one module.

2.3.2 Logic

Please read the description of the following five basic parts of the role based access control of OpenCA carefully. They explain the whole concept which is used for OpenCA. We don't use attribute certificates because they are not supported by the todays software.

2.3.2.1 Roles

The roles are part of every role based system. OpenCA defines a set of default roles which you can simply extend by other roles which you need. Every certifcate will be assigned a role if it is issued on the CA. The role of a certificate service request is the role which the requests asks for. The role of a certificate revocation request is the role of the certificate to which the CRR belongs. The CA-certificate(s) and the CRLs have no explicit role because they have automatically the *"superrole"*. If there is an action where the user is not identified by a certificate then the role which is used is automatically the empty role. This is sometimes necessary for example if you want to control your public gateway by RBAC.

2.3.2.2 Modules

Every installed gateway of OpenCA is a module in the terms of RBAC. If you install the RA then there is a new module with the name "RA 1". The details of the configuration are described later. The access rights must be defined for every module again (except they have all the same name).

2.3.2.3 Operations

An operation is a type of action which can be done on the system. An operation can be "csr approve" which means that a certificate service request will be approved. These are only logical operations.

2.3.2.4 Scripts

The scripts are the files which are placed in lib/cmds/. Every script has a configurationfile which contains the name of the command (this is actually a protection against the renaming of files), the name of the operation for which it is used, the way how to find the affected object and the name of the variable which contains the data which is necessary to determine the object by the specified way.

2.3.2.5 Rights

The rights build the ACL for OpenCA. An access right consists of four things:

- the module which is used
- the operator which works on the module (the operator is a role)
- the operation which will be performed
- the owner of the affected module (the owner is a role)

2.3.3 Technical details

2.3.3.1 Algorithm

The different configurationfiles are stored in etc/rbac/

- modules/
- operations/
- rights/
- roles/
- scripts/

The files in modules/ contains nothing. The decoded filename is the name of a role. The files in operations/ and roles/ are used in the same way like the files in modules/.

The files in scripts/ contain four variables:

- SCRIPT
- OPERATION
- OWNER_METHOD
- OWNER_ARGUMENT

There are six OWNER_METHODs:

1. CERTIFICATE_SERIAL
   This method is used if an operation affects a certificate and the role should be detected by the serial of the certificate.
2. REQUEST_SERIAL
   This method is used if an operation affects a CSR and the role should be detected by the serial of the CSR.
3. CRR_SERIAL
   This method is used if an operation affects a CRR and the role should be detected by the serial of the CRR. The CRR will be loaded and the certificate which should be revoked will be loaded and the role of the certificate is used.
4. CGI
   The use of this method is not recommended because the role is not protected by any cryptographic mechanisms.
5. ANY
   The operator must have the right to perform this operation for every role.
6. <empty>
   This method is used to signal that an object is handled which affects the CA directly (e.g. CA-certificate, CRL). The operator needs access to the *superrole*.

The configurationfile end with the suffix ".conf". A second file with the suffix ".sig" contains a signature of the configurationfile to protect the file against manipulation during transport.

The files in rights/contains the signature of the filename to protect the right. The filename itself consists of the for encoded parameters module, operator, operation and owner. The diffent parts of the filename are seperated by a single "-".

2.3.3.2 Digital Signatures

Actually there is a discussion about the signing of the files in scripts/ and rights/ because the signing costs a lot of performance and we don't really improve the security with the signing.

2.3.3.3 Modified Base64

We use a modified Base64-encoding for the filenames. We replace "/" by "_". This was necessary because Unix interprets "/" as a directorychange and the escaping of such special characters is different on several platforms.

Chapter 3
Operator Guide

3.1 Introduction

Here we have to include lifecycle.ps if somebody knows how to do this with TeXmacs.

3.2 Initialization

The initialization consists of three phases. The first phase initializes the CA itself, the second phase creates the first user-certificate and the last phase creates the first certificate for a web-server.

3.2.1 Initialize the certification authority

The following steps are necessary to setup the PKI, so not all steps are cryptooperations.

1. DB Setup
   Here you have only to click on the link and then the configured database will be initialized. This action will remove all data from the database if the database already exists. (Only SQL-databases are protected against the destruction of an existing database.)
2. Keypair Setup
   This is the first cryptooperation. You can choose the length of the private key and you must enter the symmetric encryptionalgorithm and the passphrase to encrypt the new private key. The private key is an RSA-key. We don't support DSA today.
3. Request Setup
   The CSR for the CA-certificate can be generated by this link or you can create the request by hand. If you use the link then OpenCA will ask you for the components of the CA's distinguished name and the passphrase of the CA's private key to create the CSR.
   If you want to create the request by hand then you must simply create the request and store them in OPENCADIR/var/crypto/reqs/careq.pem.
4. Certificate Setup
   The setup of the CA-certificate allows to general ways. You can create a self-signed CA-certificate if you want to create a Root-CA or you can export the CSR and import the CA-certificate if your CA is a sub-CA.
5. Final Setup
   The last step is used to create the certificate chain which is necessary for the verification fo digital signatures and to export the whole configuration. This configuration you must import into the RA before you start using the RA in production.

3.2.2 Create the initial administrator

The initial administrator certificate is the first user-certificate. You should use the role CA Operator for this certificate because this is the certifcate of the operator which issues the first certificates.

1. Create a new request

Simply enter all the data which are requested and decide which keylength you want. Please notice that many hardware security modules (e.g. several smartcards) only support keys with a maximum length of 1024 Bit. If you want to store the private key and the certificate of the initial CA Operator on a smartcard please check the available keysizes before you get into trouble.

2. Edit the request
   Now you have the last chance to modify the distinguished name of the certificate. You can fix the role and set the correct Subject Alternative Name. The DN uses the order which is described in RFC 2253.

3. Issue the certificate
   Simply click on issue Certificate to create the new certificate. You can use the button on the page which is displayed after the editing of the request too.

4. Handle the certificate
   This page is also available via Information –> Certificates –> Valid Certificates. Here you can download the certificate and the private key.

3.2.3 Create the initial RA certificate

The initial RA certificate is the certificate for the https-server. You should use the role Web Server for this certificate.

The steps are the same like for the initial administrator.

3.2.4 Troubleshooting

If I click on edit request then I see an errormessage.

If the errormessage is *Request not present in DB or the status of the request was changed!* then you start the intialization without a re-intialization of the database. The initial request has not the serial 1 and the link is wrong. You must start again with the initialization of the database but you must not create the private key and the request for the CA again.

If I click on issue Certificate then I see the errormessage *pwd.html not found*.

This message appears if you configured the wrong path for the apache's DocumentRoot. This can happen in httpd.conf or by using a wrong value during ./configure. Please contact your administrator or read the administratorguide for more information.

If I click on issue Certificate then it failed.

Please check your Apache's error.log. Does your errormessage looks like the following one?

```
31839:error:2206D06C:X509 V3 routines:X509V3_parse_list:invalid null
name:v3_utl .c:319:

   31839:error:2206B069:X509 V3 routines:X509V3_EXT_conf:invalid extension
   string:v 3_conf.c:138:name=subjectAltName,section=

   31839:error:2206B080:X509 V3 routines:X509V3_EXT_conf:error in
extension:v3_conf .c:92:name=subjectAltName, value=
```

This means that you forget to edit the request. OpenCA doesn't set the subject alternative name automatically. It only gives you a recommendation. You must press at minimum ok. If you don't want to set a subject alternative name then you must change the configurationfile of the extensions of this role (OPENCADIR/etc/openssl/extfiles/your_role.ext).

I exported the configuration (or all) from the CA and my certifcates are deleted.

This is normally not the exact discription. You exported the configuration (or simply all) and then you start initializing the RA but the CA and the RA are on the same machine with the same configuration. The first step you are doing is initialize database.

This will initialize the database but the CA and the RA use the same database. If you use DBM-files (DBmodule "DB") then you overwrite in this moment the CAs database too. If you use a SQL-database (DBmodule "DBI") then nothing is happen and you get only an errormessage that the initialization on the RA fails.

I try to export my configuration and the export fails.

The permissions of /dev/fd0 are the most common problem. The most Unixsystems change the owner of /dev/fd0 at every time a user logged in on the console. So you should check the owner and rights of /dev/fd0.

Cannot convert PEM-certificate and PKCS#8-key to PKCS#12-formatted file!

This messages occurs if you try to export a private key and the associated certificate to a PKCS#12-file. You must use the passphrase which you entered in the two PIN-fields during you make the request. You must not enter the CA's passphrase here!

3.3 Export and Import of data

The export and import of data is necessary to exchange the data between the CA which is offline and the rest of the trustcenter. Actually we support the fillowing functions:

1. From CA to RA
   - CA-Certificate(s)
   - Configuration

     These functions supports the exchange of the CA-certificates and the RBAC-related data.
   - Certificates
   - CRLs
   - All

     This functionality should normally used. It exports and imports all relevant data. So you must not care about anything by yourself.

2. From RA to CA
   - CSRs
   - CRRs
   - All

     This functionality should normally used. It exports and imports all relevant data. So you must not care about anything by yourself.

The main problem today is that we must export and import all data at every time because there is no verification mechanism for the export and import which checks the successful transport of objects.

3.4 CSR handling

The handling of a certificate service request consists of four steps:

1. Editing

2. Approving
3. Export/Import
4. Issuing the certificate

3.4.1 Editing the CSR

3.4.1.1 Distinguished Name

The distinguished names which are displayed and sometimes editable in OpenCA uses the order defined in RFC 2253. All conversions are handled automatically. A sample DN could be "cn=John Doe, ou=public relations, o=Joe's Pub, c=uk".

3.4.1.2 Subject Alternative Name

The subject alternative name is the place where the emailaddress or the DNS-name can be stored. OpenCA uses the emailaddress by default because the emailaddress should not be part of the DN of a usercertificate (see RFC 2633 "S/MIME Version 3 Certificate Handling" section 3).

The subject alternative name must use the format defined by OpenSSL. The format looks like follows:

```
SubjectAltName ::= GeneralNames
GeneralNames ::= GeneralName ["," + GeneralName]*
GeneralName ::= CHOICE {
        rfc822Name,
        dNSName,
        uniformResourceIdentifier,
        iPAddress,
        registeredID}
rfc822Name ::= "email:" + IA5String
dNSName ::= "DNS:" + IA5String
uniformResourceIdentifier ::= "URI:" + IA5String
iPAddress ::= "IP:" + OCTET STRING
registeredID ::= "RID:" + OBJECT IDENTIFIER
```

Perhaps you like examples more then pure definitions:

1. email:copy,email:my@other.address,URI:http://my.url.here/
2. email:my@other.address,RID:1.2.3.4
3. email:admin@myhome.net,DNS:https://www.myhome.net,IP:123.123.123.123

The first example copies the emailaddress from the subject. The subject is the distinguished name of the certificate.

3.4.1.3 Role

The role is important for the extensions and for the logical meaning of the certificate.

3.4.2 Approving the CSR

You must simply view at the request and if you think that all relevant items are correct then you can use one of the approve buttons. If you use Approve and Sign then you can sign and approve the request with a Netscape 4.7x or IE 5.0+. If you cannot or must not sign the request then you can use the button Approve without Signing. The data of the request cannot be changed after the request was approved.

3.4.3 Export/Import of the CSR

If you use an offline-CA then you must export the requests from the RA and import the requests into the CA. More details you can find in the section *"Export and Import of data"*.

3.4.4 Issuing the certificate

Before you issue a certificate on the CA please check the displayed data carefully. This is the last chance to delete the request. If you want to issue the certificate simply click on Issue certificate and enter the passphrase of the CA. The new certificate will be displayed.

3.4.5 Renew a CSR

This is actually not implemented.

3.4.6 Troubleshooting

General Error Trapped 6203: The request is not signed!

This error can be caused by several problems. Here the most common ones:

1. You did not import the CA-certificate into your browser. This can only happen if you created the request for your certificate with your browser and imported the certificate via the public gateway of OpenCA. If you generated the request with the form "Basic Request" and you get a PKCS#12-file from OpenCA to import it into your browser then you have the CA-certificate already.
2. Your browser know the CA-certificate of your certificate but you don't trust this CA. If you want to sign with your certificate then you must trust the CA.

General Error Trapped 6206: Cannot build PKCS#7-object from extracted signature!

This can happen if the databases of your browser which store the certificates and keys are corrupt. This is possible if you are adding a new CA-certificate but use the same name like for an already existing CA-certificate (e.g. you made your second testinstallation but forgot to remove the old CA from your browser).

The second possibility is that you have not imported the CA-chain. If you imported the hole chain then you must trust at minimum the root-CA. Netscape doesn't accept untrusted CA-certificates for signing and Netscape needs the hole CA-chain for signing.

You have problems with IE and signing?

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/security/security/capicom_start_page.asp
If you need a downloadaddress for CAPICOM then you should try this:
http://www.microsoft.com/msdownload/platformsdk/sdkupdate/default.htm?p=/msdownload/platformsdk/sdkupdate/psdkredist.htm

Cannot encrypt PIN-mail! Aborting!

This problem can have two reasons. The encryption of the PIN-Mail can fail if you use some really unstable OpenSSL-snapshots but the common mistake is an unconfigured SERVICE_MAIL_ACCOUNT. You can configure the mailaddress in ca.conf.

3.5 CRR handling

The general way for a CRR is the same like for a CSR. Therefore we only discuss the troubleshooting here.

The only different between a CSR and CRR is that an Operator can start a revocation on the RA. If OpenCA view a certificate and you see the button revoke then you can start a revocation.

3.5.1 Troubleshooting

General Error Trapped 6206: Cannot build PKCS#7-object from extracted signature!

This can happen if the databases of your browser which store the certificates and keys are corrupt. This is possible if you are adding a new CA-certificate but use the same name like for an already existing CA-certificate (e.g. you made your second testinstallation but forgot to remove the old CA from your browser).

You have problems with IE and signing?

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/security/security/capicom_start_page.asp

3.6 CRLs

Here we have to insert some data ...

3.7 Batchprocessors

OpenCA has several very powerful batchprocessors to support big organizations with certificates and don't causes a lot of money. There are two categories of batchprocessors. One class of batchprocessors should automate the work of the CA Operators and the other class of batchprocessors should automate the work of the RA Operators.

Nevertheless all batchprocessors are installed on the CA by default because the use of nearly fullautomatic software is a high risk.

3.7.1 CAO batchprocessors

3.7.1.1 Issue certificates

If you have a big organization with many RA Operators which uses their certificates and private keys to approve and sign the certificate requests of your members then it is perhaps not necessary to have a CA Operator whom checks every request again.

The main idea is that the CA Operator only enters the following data:

- the role of the RA Operator(s)
- the requested role of the certificate request
- the passphrase for CA's private key

The batchprocessor issue every certificate from a request where the requested role matches the specified role, the signing RA Operator has the specified role and the signature is correct.

The usage of this automatic system is not a high risk if you can trust the signatures of your RA Operators.

3.7.1.2 Revoke certificates

The main difference between this and the above described processor is that this processor uses the role of certificate which is assigned to the processed CRR.

3.7.2 RAO batchprocessors

These software components has a complete other focus than the CAO batchprocessors. The idea is that you have some ERP-like systems in the background and by this way you have a defined base of users. Now you need a simple way to enroll a PKI.

Below we describe some batchprocessors but they are only examples. You can find the code in your OpenCA-directory (default: /usr/local/OpenCA/lib/cmds/bp*). If you want to implement another businessprocess simply do it.

3.7.2.1 New users

Let's start enrolling a PKI!

Import new users

First we must import the new users so we need a file on the standard importdevice. The name of this file is specified in /usr/local/OpenCA/etc/servers/ca.conf and the option is BP_File_ImportNewUser. This file must have the following format

```
ID 1234567890
DN CN=testuser, OU=Research, O=OpenCA, C=DE
ROLE User
SUBJECT_ALT_NAME email:testuser@openca.org
STATUS_MSG Does somebody know this guy?
<blank line>
<next dataset>...
```

The ID is a freestyle but unique ID generated by your ERP- or IT-systems. It is required. The DN is an OpenSSL-like DN which is used for the certificate request. OpenSSL-like mean that the attributes must be conform to the definitions in include/openssl/objects.h. Please take in mind that OpenSSL is **casesensitive**. The DN is required. The role specifies the role of certificate which should be issued for the user. It is required.

The SUBJECT_ALT_NAME is OpenSSL-style string which conatins the setting for subjectAltName in openssl.cnf. So if the specified role requires a subject alternative name then you must specify this option.

All other added options are stored in a file. So you can find you message in the file /usr/local/OpenCA/var/batch/1/2/3/4/5/6/7/8/9/0/STATUS_MSG.

A empty line signals a new dataset.

Import permissions

You can import all the necessary permissions with this command. The option in ca.conf is BP_File_ImportACL. A permission has the format "PERMISSION_NAME (grant|deny)". Following you can find an example:

```
ID 1234567890
newPIN grant
newKey grant
backupKey grant
newCSR grant
approvePendingCSR grant
<blank line>
<next dataset>...
```

Create PINs

Now we need some PINs for our new users to be able to handle their certificate requests automatically.

Export PINs

The generated PINs must be exported to your system to allow you to send some PIN-mails. The exported file has the format:

```
ID 1234567890
PIN kasldfjhfhkljahaf2e32
<blank line>
<next dataset>
```

You can delete the PINs after you send the mail because the CA has a copy of the PIN and the hashed PIN.

Approve requests
The user go with his PIN and his ID to a terminal and requests a certificate via the public gateway. He must enter the PIN in the PIN-field and the ID into the field for the name of the user. The batchprocessor will check the ID, PIN and requested role, add the correct DN and if specified the subject alternative name and sign the hole request.

Delete not hashed PINs
After a request was approved we need no longer the PINs. So simply remove them.

It is possible to mix the steps because they are safe against such *mistakes*. Did you forget 100 users? No problem, simply build a new file and start again (at minimum you must do the steps again which you don't perform for the other users).

3.7.2.2 Renewed certificate requests

Sometimes there are users which need a certificate renewal (e.g. the certificates for the students of your university are only valid for one year). So what should you do if you must do so many renewals?

Import permissions
You must build a file with the name of the option BP_File_ImportACL in ca.conf. The file has the same format like for **Import Permissions** for new requests.

Renew requests
The requests will be automatically renewed if the correct permission is set.

Approve renewed users
Now you must simply do the same like for **Approve request**.

3.7.2.3 Update data

Sometimes there are users who want to marry. The problem is that you need the man or woman and cannot forbid them to marry. So you must update there data. No problem. simply build a file like for **Import new users** and this batchprocessor updates the data of such persons. If you now start a renewal for this user then he will get a new certificate.

3.8 Backup and Recovery

We must not discuss the necessity of backups. So we can start with the details and we start with a warning:

OpenCA create no backups of the private key!

If you create a backup with OpenCA then all available cryptoobjects in the database will be exported from the database. The backup will be written to the standard export-device of OpenCA.

If you have a crash and you want to recover then you must first restore the private key and then you must know what do you use for a database. If you use DBM-Files (the default) then you must use the importDB-functionality of OpenCA. This will restore all the objects in your database. If you use a SQL-database then you must use the function replayLog. The SQL-databases include a log where all write-actions are documented so OpenCA can replay all actions which ever happens to the module. The function importDB works for SQL-databases too but replayLog is much better.

Chapter 4
User Guide

4.1 CA-Certificate

Before you can start using a PKI, you must know whom you can trust. The basic idea of a PKI is that all users trust an authority. This authority create a so called CA-certificate. This is the root of all trustrelationships.

If you take this in mind then the first step of you in a PKI-environment is the download of the CA-certificate. So let's do it :)

4.2 Certificate Service Requests

OpenCA knows several different types of requesting a certificate. Perhaps your administrator configure a different one. Here we explain only the default types.

4.2.1 User or client requests

If you need a certificate for your browser or your emailclient then you can choose between three methods - SPKAC for Mozilla and Netscape, PKCS#10 for Microsoft Internet Explorer and BASIC for all other clients. If you are not sure what you should use then simply use the link for the automatic browserdetection. OpenCA is able to detect your browsertype and will present you the correct form.

If you want to ask your administrator for a smartcard then please use the token request.

4.2.1.1 Mozilla und Netscape

If you need a certificate for your browser or messenger then this is the right way. You have only to enter the data. If you finished and pressed "continue" then you will see your data again. Please check it carefully. If you now press "continue" again then Mozilla starts the keygeneration.

4.2.1.2 Microsoft Internet Explorer

If you need a certificate for a Microsoftapplication then you should use this form. The Internet Explorer will start the keygeneration after you pressed continue for the second time. The Internet Explorer displays during the process some informal messages. This includes a notice about the patchlevel of your browser. This is done because Microsoft found a bug in an Active-X Control which is necessary for the certificate enrollment. The bug is security relevant.

4.2.1.3 Other Clients

This is the easiest way for you to request a certificate. You can use every browser to do this including Opera and Konqueror (and Lynx too for the really tough guys). You must only fill out the form and check your inserted data.

Please memorize (and not notice) the entered PIN. This PIN is necessary to convert the created key into a format which your browser understand. No human or system except of you know this PIN after the requestgeneration is completed.

If you finished then go to the RA of your PKI or follow the displayed instructions.

4.2.1.4 Token request

This is not a request in the meaning of PKI. The form will only create a header which signals the operators that you want hardwaretoken. All cryptographic operations will be performed by the operators.

4.2.2 Server requests

If you have web server, mail server, ldap server or VPN server then you need perhaps a certificate. There are two ways to get such a certificate - you can generate a private key and a request by yourself or we do this job for you. If you want to create the private key and the request by yourself then please use the link for server requests (PKCS#10 requests). If you want that we do all things for you then use the link for a "Basic request".

4.2.2.1 PKCS#10 requests

If you use this form then you must upload your request and enter some additional data. OpenCA simply stores your request and the additional data in it's database.

4.2.2.2 Basic request

If you use this method then please don't use the link for the automatic browserdetection. You must use the link "Basic request". When you entered all your data and checked them again then OpenCA will generate a request and private key for you. It's not necssary that you learn how OpenSSL works but if you want to be a serious administrator then we can only recommend you to understand how to generate a private key and a request.

4.3 Certificate Revocation

4.3.1 Where can I revoke a certificate?

There are three different ways how a user can start a revocation. The first and simplest way is to go to the public interface, click on "Revoke Certificate" in the navigation bar and enter the serial number of the certificate in the form. The second method is simple too but it can take a longer time. Go to the certificate lists, choose a certificate and click on "Revoke the certificate". The last method is perhaps the most comfortable method. Got to the certificate search, search the certificate, choose the correct certificate and click on "Revoke the certificate".

4.3.2 How do I fill the revocation form?

First of all please enter a reason. This is really important to give the operators an idea how fast they have to react. If you only forgot your passphrase then they can revoke the certificate one or two days later. If somebody steal your smartcard and perhaps you cannot find the letter with the passphrase inside then they have to revoke the certificate really fast to avoid attacks.

The second thing is the authorization of the revocation. You can choose between two methods. You can use the CRIN-code which you received in an encrypted mail. This CRIN-code is a very long and secure passphrase. Simply enter the code and send the form without signing. If you don't have the CRIN-code but you have a certificate and a private key then you can ignore the CRIN-code in the first form, click on "continue" and then send the second form with a digital signature (click on "Sign and submit").

If you have no CRIN-code and you have no certificate and private key then please go to your nearest registration authority. There are operators which can start the revocation.

4.4 Certifcate Revocation Lists

CRLs are used to publish the serials of the certificates which should no longer be used. Please download an actual CRL at every time you use the PKI. Modern browsers like Mozilla support automatic updates. You must only go to the webpage with the CRL and click on it then Mozilla will ask you for automatic updates of this CRL. This is much simpler then manual updates which you normally forget ;-)

Glossary

ACL	Access Control List	33
PKI	Public Key Infrastructure	43
CA	Certification Authority	43
RA	Registration Authority	43

Appendix A
Authors

Michael Bell <michael.bell@web.de>

Chris Covell <chris@katjam.co.uk>
LDAP documentation in the Administrator Guide

Harald Wallus <wallus@results-hannover.de>
Example installation of OpenCA