GDigiDoc

Users Guide

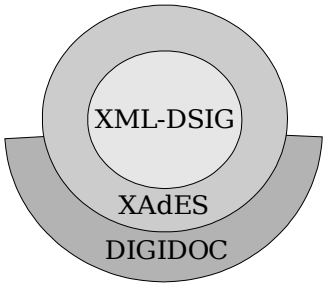
Veiko Sinivee

Table of Contents

Concepts

GDigiDoc is a GUI program for creating, editing and verifying digitally signed documents. The name GDigiDoc stands for "**G**NU **Digi**tally signed **Doc**uments". GDigiDoc is based on the DigiDoc (also known as OpenXAdES) C library. More information on this library can be found on OpenXAdES website. DigiDoc files are XML files and the signatures follow XML-DSIG and XAdES (ETSI – TS 101 903) specifications. However one can sign any kind of data by including it in Base64 encoding in an XML document. A digitally signed document is actually a container containing one or more data files and one or more signatures. All signatures must sign all data files. Thus after adding the first signature you can no longer add, modify or remove data files. A data file can contain the original file directly (if it was an XML or text file), in Base64 encoding (if it was in some binary format) or a reference to an external "DETATCHED" file.

This library supports only OpenXAdES file format. OpenXAdES format is based on XAdES and adds a general XML container. This enables the library to create digitally signed documents, that contain both many data items and many signatures.



The library supports RSA-SHA1 signatures. DigiDoc library includes functions for accessing smartcards for signature operations using a PKCS#11 driver. Another version of this library is available on Microsoft Windows platforms in form of a COM component - DigiDocLibCOM. The latter uses Microsoft CSP API for accessing smartcards. There is also a version of this library - JDigiDoc - written in the Java programming langage.

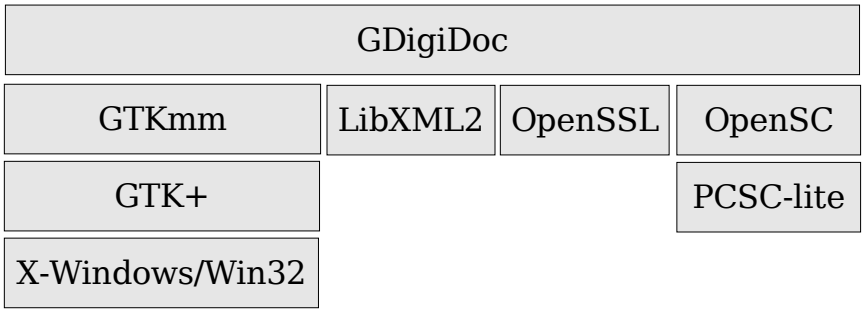
In order to use this application you must have a smartcard with some RSA keys and a card-reader attached to your computer. You need also drivers for your card reader and PKCS#11 driver for your card type. How to install these is documented below.

In Estonia digital signatures are considered equal to handwritten signatures. However if longtime proof and validity is requred, then one must add an OCSP confirmation to the signature. This OCSP is a signed confirmation that the signers

certificate was valid at the time of signing. OCSP confirmations are issued to all Estonian ID cards. If you don't have one then you can still use the program by adding your certificate to Estonian OCSP responders demo version here: http://www.openxades.org/tryitout.html. Such confirmations are issued by the demo-responder and they have no legal status but this enables you to try the software. Offcourse you can also use the software to read existing digitally signed documents. You don't need a cardreader and smartcard for that.

Installation

GDigiDoc uses GTKmm API for the popular GTK+/GNOME environment. It will work also under KDE, but you must install GTK+ and GTKmm. GDigiDoc uses libdigidoc library for handling digitaly signed documents. This library in turn uses OpenSSL for cryptography, LibXML2 for XML parsing and OpenSC for smartcard access. OpenSC library in turn needs the PCSC-lite smartcard daemon.



GDigiDoc source code can be downloaded from the projects download page at SourceForge.net: http://sourceforge.net/projects/gdigidoc.

Installing GTKmm

Installation instructions for GTKmm can be found here: http://www.gtkmm.org/download.shtml. There are binary packages for Debian and also for RPM based distributions like RedHat and Mandrake. Unofficial RPM-s for Fedora Core2 can be found here: http://www.poolshark.org/fc2.html. Binary packages for Win32 environment can be found here: http://www.pcpm.ucl.ac.be/~gustin/win32_ports. Offcourse you can also download the source from GTKmm home and compile it but that takes a lot of time.

Installing PCSC-lite

Download the newest PCSC-lite from here: https://alioth.debian.org/project/showfiles.php?group_id=1225.

Install it like that:

```
# ./configure
# make
```

```
# make install
```

This installs the **pcscd** daemon to /usr/local/sbin and the libraries to /usr/local/lib. You must add /usr/local/lib path to /etc/ld.so.conf if you haven't done it already and then run **ldconfig**. An alternative would be to use

```
# ./configure –prefix=usr --sysconfdir=/etc
```

Then you need to create startup skripts and register pcscd to start automatically when you start the computer. PCSC-lite daemon can't do anything without a driver for your card reader. You can probably find a driver for your card reader from here: http://www.linuxnet.com/sourcedrivers.html. There is also an earlier version of PCSC-lite and RPM-s for RedHat and Mandrake. For some readers you can find binary packages but most come in form of source code. Don' forget to register the card reader in /etc/reader.conf. Some companies have developed Linux drivers for their card readers and you can download the driver from the companys website. Here are some sites where you can find more info about cardreaders and Linux:

- http://www.konsultant.ee/mod.php?mod=userpage&menu=110101&page_id=17
- http://www.id.ee/pages.php/030211?foorum=3
- http://martin.paljak.pri.ee./esteid/

Installing OpenSC

PCSC-lite and cardreaders driver provide means to communicate with the card reader device. So you can send and read data to/from any cards in the device. In order to have the smartcard execute some command you have to know it's command syntax. This is what a PKCS#11 driver does for you. It creates an abstraction layer that enables a programm to communicate with any smartcard provided that you have a PKCS#11 driver for this card. For Estonian and Finnish ID cards one can use the OpenSC library for this purpose (http://www.opensc.org). A specific version of this library for Estonian ID card is available here: http://martin.paljak.pri.ee./esteid/. You can find here also instructions how to install OpenSC and test it. Download the source code and do:

```
# ./configure --prefix=/usr --sysconfdir=/etc --with-pcsclite=/usr
# make
# make install
```

Make sure you register the directory with PKCS#11 driver (e.g. /usr/lib/pkcs11/ or /usr/local/lib/pkcs11/) in /etc/ld.so.conf and run ldconfig. Then you can test it by registering opensc-pkcs11.so as a new security device driver in Mozilla/Netscape and try accessing some website that requires authentication. For example https://sk.ee/cgi-bin/tervitus. You can find links to websites using smartcard authentication here: http://www.id.ee/pages.php/030207,157.

Installing libdigidoc

GDigiDoc uses libdigidoc. Grab the newest libdigidoc from the same site you

downloaded gdigidoc (e.g. http://www.sourceforge.net/projects/gdigidoc). Install the library like that:

```
# ./configure --prefix=/usr --sysconfdir=/etc
# make
# make install
```

This library comes with a small testprogramm – digidoc – that provides all the same functionality as gdigidoc, except that it's a command-line program. This utility helps you to verify that libdigidoc has been properly installed. Using and configuring libdigidoc has been documented below.

Installing GDigiDoc

Download gdigidoc's latest version from the projects site at SourceForge.net (http://www.sourceforge.net/projects/gdigidoc) and install it like that:

```
# ./configure --prefix=/usr --sysconfdir=/etc
# make
# make install
```

Then you could create a launcher for GDigiDoc on your GNOME panel and use the provided id-logo.png for icon.

Using GDigiDoc

If you succeded in installing PCSC-lite, cardreader driver and OpenSC, which you could verify by registering the PKCS#11 driver in Mozilla, then it's now time to check if libdigidoc has been properly installed and then try using gdigidoc.

Configuring libdigidoc

Libdigdoc uses the global configuration file - /etc/digidoc.conf and a user-specific configuration file ~/.digidoc.conf. In the global configuration file you can find the following entries:

```
# CA certificates – configured properly for Estonian ID cards.
CA_CERTS=4
CA_CERT_1=/usr/local/share/certs/JUUR-SK.PEM.cer
CA_CERT_1_CN=Juur-SK
CA_CERT_2=/usr/local/share/certs/ESTEID-SK.PEM.cer
CA_CERT_2_CN=ESTEID-SK
CA_CERT_3=/usr/local/share/certs/TEST-SK.PEM.cer
CA_CERT_3_CN=TEST-SK
CA_CERT_4=/usr/local/share/certs/KLASS3-SK.PEM.cer
CA_CERT_4_CN=KLASS3-SK
CA_CERT_PATH=/usr/local/share/certs
```

```
# Default file format and version
DIGIDOC_FORMAT=DIGIDOC-XML
DIGIDOC_VERSION=1.3

# Usable PKCS#11 drivers and the default driver
DIGIDOC_DEFAULT_DRIVER=2
DIGIDOC_DRIVERS=2
DIGIDOC_DRIVER_1_NAME=EYP driver
DIGIDOC_DRIVER_1_DESC=Eesti Yhispanga loodud ID kaardi PKCS#11 ohjurprogramm
DIGIDOC_DRIVER_1_FILE=/usr/local/lib/esteid-pkcs11.so
DIGIDOC_DRIVER_2_NAME=OpenSC
DIGIDOC_DRIVER_2_DESC=OpenSC baasil loodud PKCS#11 ohjurprogramm
DIGIDOC_DRIVER_2_FILE=/usr/local/lib/pkcs11/opensc-pkcs11.so
# default slot id for signature key – configured for Estonian ID cards!
DIGIDOC_SIGNATURE_SLOT=1

# OCSP responders certficates – configured for Estonian ID cards and “demo-
responder”
DIGIDOC_OCSP_RESPONDER_CERTS=3
DIGIDOC_OCSP_RESPONDER_CERT_1=/usr/local/share/certs/ESTEID-SK OCSP
RESPONDER.PEM.cer
DIGIDOC_OCSP_RESPONDER_CERT_1_CN=ESTEID-SK OCSP RESPONDER
DIGIDOC_OCSP_RESPONDER_CERT_1_CA=ESTEID-SK
DIGIDOC_OCSP_RESPONDER_CERT_2=/usr/local/share/certs/TEST-SK OCSP RESPONDER.pem.cer
DIGIDOC_OCSP_RESPONDER_CERT_2_CN=TEST-SK OCSP RESPONDER
DIGIDOC_OCSP_RESPONDER_CERT_2_CA=TEST-SK
DIGIDOC_OCSP_RESPONDER_CERT_3=/usr/local/share/certs/KLASS3-SK OSCP
RESPONDER.pem.cer
DIGIDOC_OCSP_RESPONDER_CERT_3_CN=KLASS3-SK OCSP RESPONDER
DIGIDOC_OCSP_RESPONDER_CERT_3_CA=KLASS3-SK
# OCSP responder URL
DIGIDOC_OCSP_URL=http://ocsp.sk.ee
# optional HTTP proxy – change this !
DIGIDOC_PROXY_HOST=proxy.yourhost.com
DIGIDOC_PROXY_PORT=8080
```

Global configuration file contains entries that are usually the same for all users of your computer. User-specific configuration file contains entries for the given user. Such entries allways override the global configuration file entries.

```
# Your PKCS#12 token file and password required to access the OCSP responder.
# you can get this PKCS#12 token from here: http://www.sk.ee/pages.php/02020504
# Change this!!!
DIGIDOC_PKCS_FILE=<pkcs12-token-filename-and-full-path>
DIGIDOC_PKCS_PASSWD=<pksc12-tokens-password>
DIGIDOC_OCSP_URL=http://ocsp.sk.ee
# Since gdigidoc stores it's configuration entries in the users configuration
# file you can find here also the following entries:
# Signers address & manifest mode: 0=ask before signing, 1=none, 2=use these values
MANIFEST_MODE=0
# default entries for signers address and manifest.
DIGIDOC_ROLE_MANIFEST=I agree with this document...
DIGIDOC_ADR_COUNTRY=Eesti
DIGIDOC_ADR_STATE=Harjumaa
DIGIDOC_ADR_CITY=Tallinn
DIGIDOC_ADR_ZIP=12918
# Flag – use HTTP proxy or not
USE_PROXY=TRUE
# Flag – sign OCSP requests or not. In general you allwas need to sign the
# OCSP requests unless you have a specific agreement with the service provider
SIGN_OCSP=TRUE
```

Digidoc utility

The libdigdoc library comes with a small utilityprogramm – digidoc – that provides access to most libdigidoc functions. You can use it to create, read, modify, verify and sign digitally signed documents on OpenXAdES format.

1. **Displaying help** – Use “digidoc -help” or “digidoc -?”
2. **Creating a new document** – Use “digidoc -new [format] [version]”. Default format and version are taken from the configuration file(s) if omitted. This command is optional. For example if you add a data-file using the **-add** command and there is currently no digidoc in session then a new digidoc is created implicitly.
3. **Adding a data file** – Use “digidoc -add <file-name> <mime-type> [<content-type>] [<charset>]”. Default content type and charset are taken from the configuration file(s) if omitted.
4. **Verifying signatures** – Use “digidoc -verify”
5. **Reading digidoc files** – Use “digidoc -in <filename>”
6. **Writing digidoc files** - Use “digidoc -out <filename>”

7. **Signing –** Use “digidoc -sign <pin> [<manifest>] [<city> <state> <zip> <country>]”. Default values for manifest and signers address are taken from the configuration file(s).
8. **Extracting a data-file** – Use “digidoc -extract <doc-id> <file-name> [<charset>] [<file-name-charset>]”.

Commands can be combined to execute more complicated tasks. For example:

- Read a digidoc file from the disk and verify all signatures:

```
# digidoc -in <filename> -verify
```

- Create a new document in 1.1 format, add a PDF file, sign it and verify

```
# digidoc -new DIGIDOC-XML 1.1 -add mydoc.pdf application/pdf -sign <pin> “I agree with the contract terms” -out mydoc.ddoc -verify
```

- Read an existing document, add a signature, verify and write in a new file

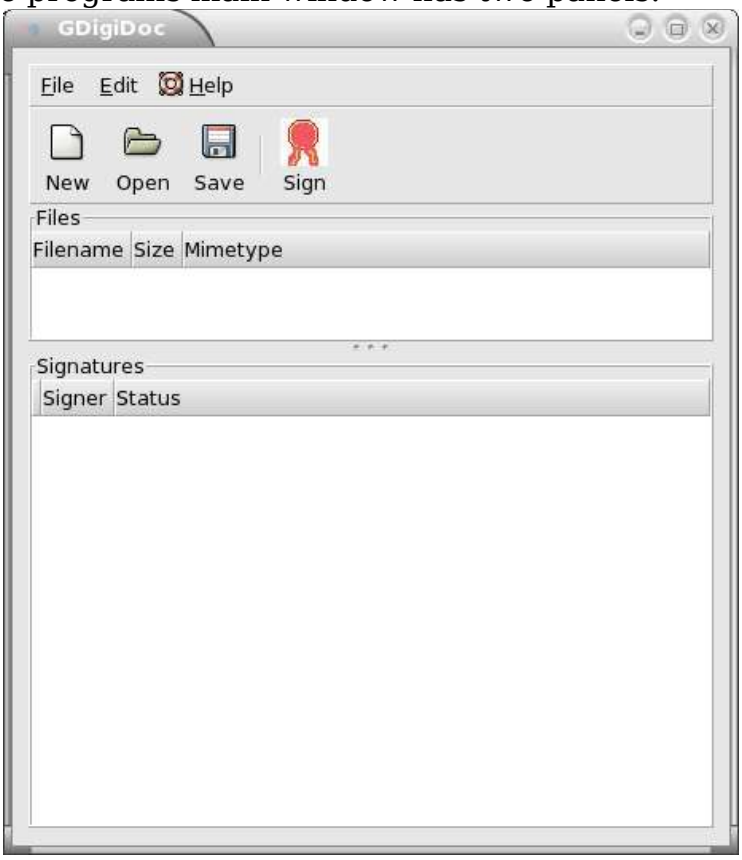
```
# digidoc -in mydoc.ddoc -sign <pin> “I reject this proposal!” -out mydoc2.ddoc -verify
```

- Read an existing document and extract one of the data-files

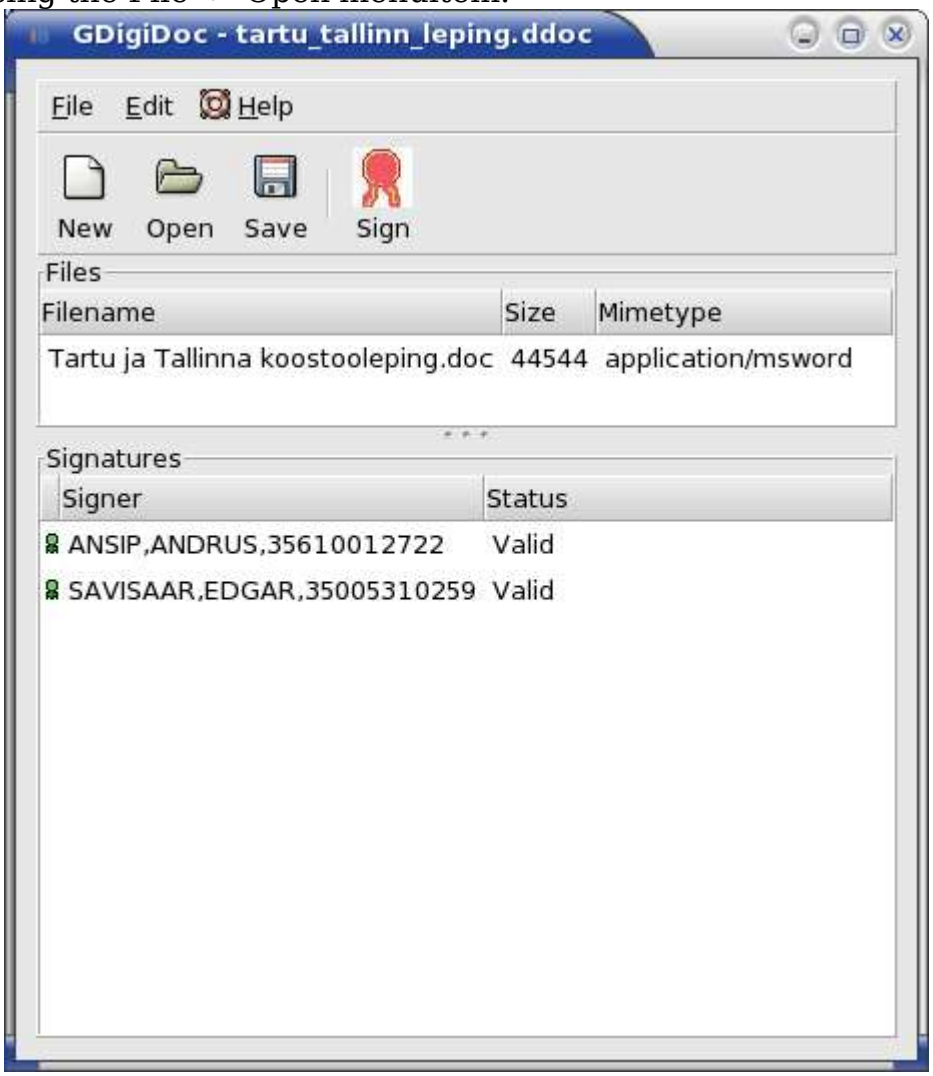
```
# digidoc -in mydoc.ddoc -extract D0 mydoc2.pdf
```

GDigiDoc

Start GDigiDoc by invoking gdigidoc or clicking on the launcher icon on your GNOME panel. The programs main window has two panels.

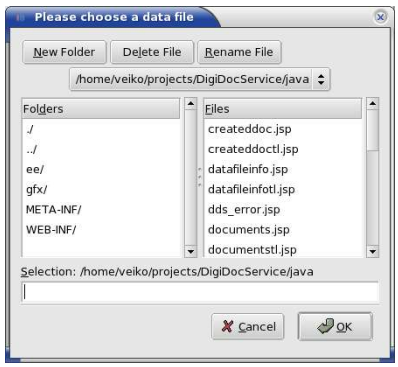


The upper panel displays a list of data-files in the current digidoc document and the lower panel displays the signatures. To view this we have the read in a digidoc document using the File -> Open menuitem.



Now we see a the name of the digidoc file on the titlebar and a list of datafiles and signatures. All valid signatures have a green icon and invalid ones have a red icon.

To create a new digidoc document, use the File -> New menuitem. This resets the lists and you can start adding new data-files. To add a data-file, select the Edit -> Add file menuitem.



The new file will be added to the list of data-files. You can remove files by selecting the desired file from the list and using the Edit -> Remove file menuitem. You cannot add or remove files if the document has been signed. Then you must remove all signatures before making any modifications to the document. Extracting data-files can be done my selecting the desired file from the list and using the Edit -> Extract file menuitem. A dialogbox will ask for a new filename and thus enables you to pick another directory where to store the file. Now you can open the file in a suitable program.

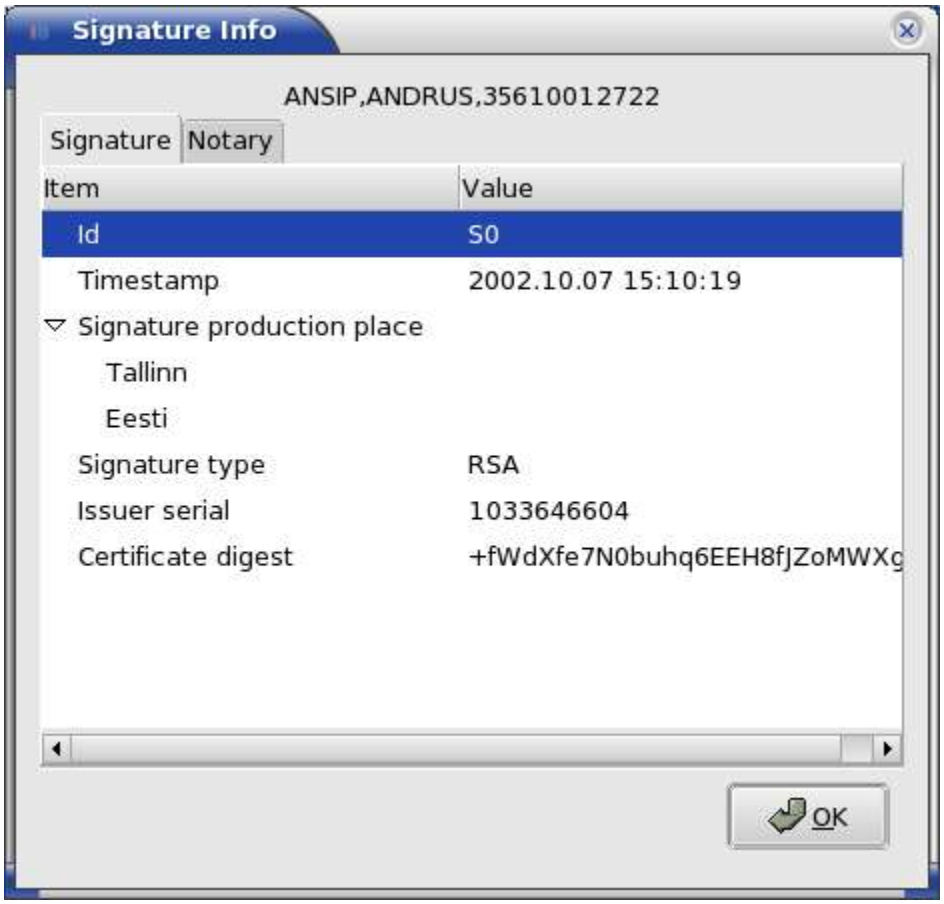
To sign the document select the Edit -> Sign menuitem or click on the Sign icon on the toolbar. Depending on the settings for signers address and manifest in the configuration file a dialogbox might pop up and display the default values for signers address and manifest.

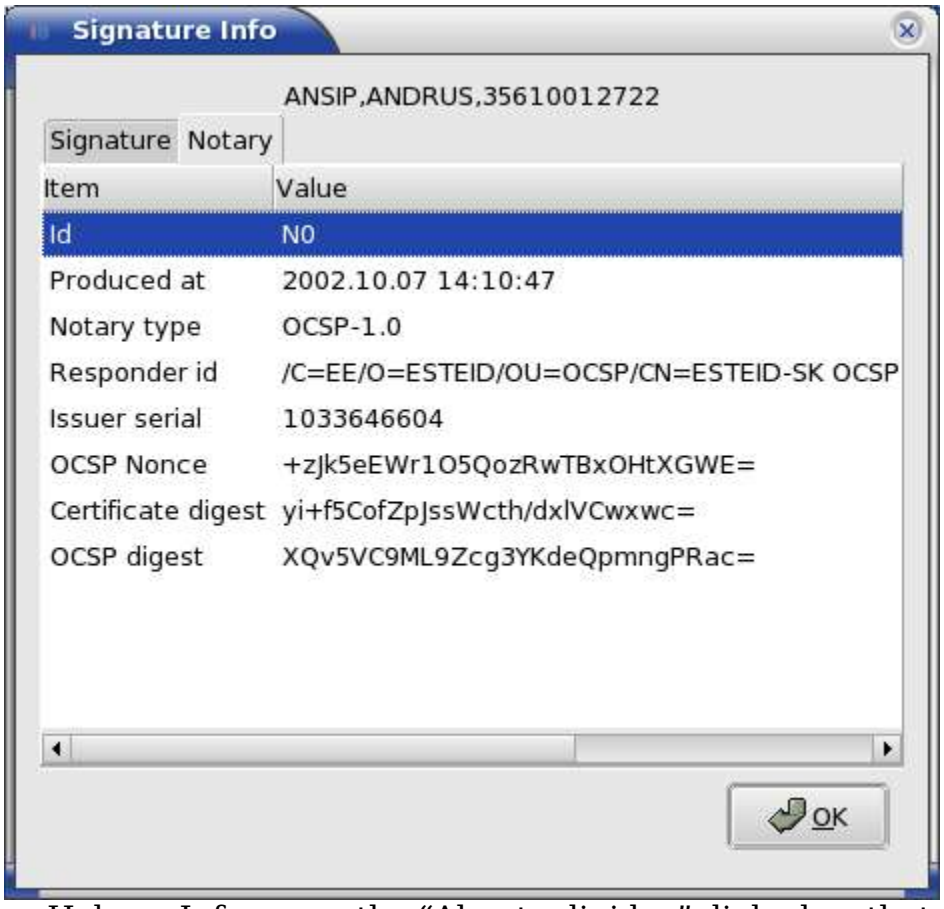
You can change the default values and continue adding a new signature or cancel the operation. Then a dislogbox will appear where you can enter your signature pin.

On Estonian ID cards this is allways the “PIN2”. GDigiDoc will now add the signature, request for OCSP confirmation and update the siganture panel. For removing the signature select the desired signature and use the Edit -> Unsign menuitem.

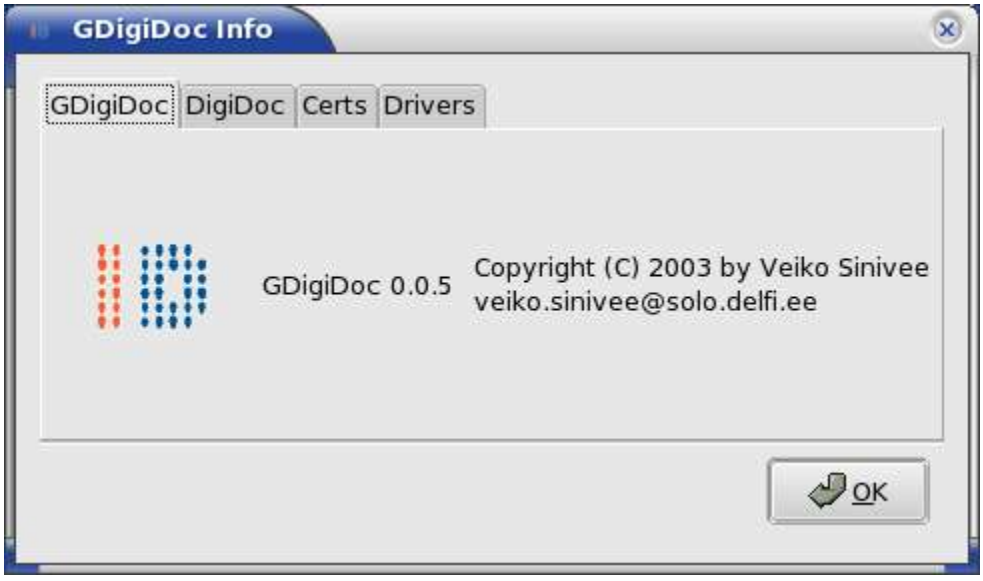
Both panels have context-sensitive menus that you can display by right-clicking on the datafile or signature item. These menus provide quick access to the corresponding Edit menu items. For example to view detailed data-file info, select the desired file from the upper panel, righ click on it and select “Info”.

The signature info dialog has two panels displaying signature and OCSP confirmation details.

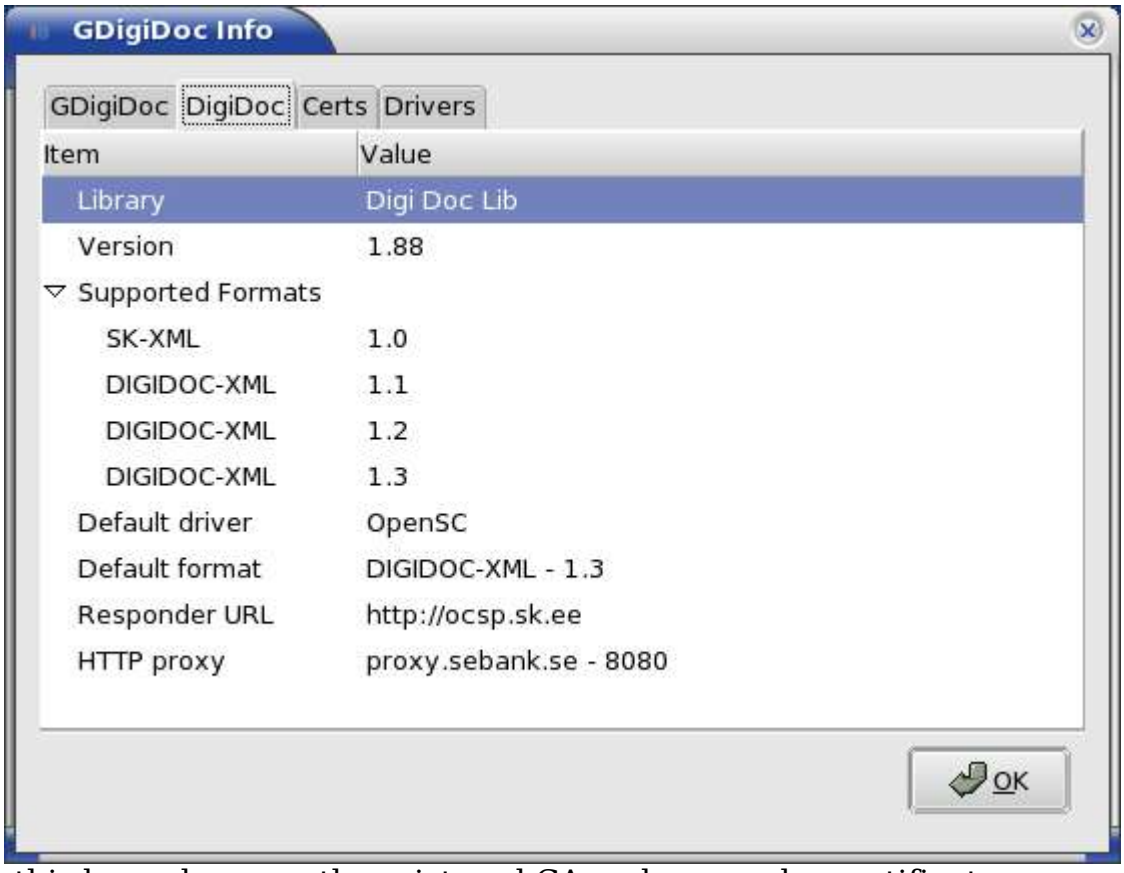




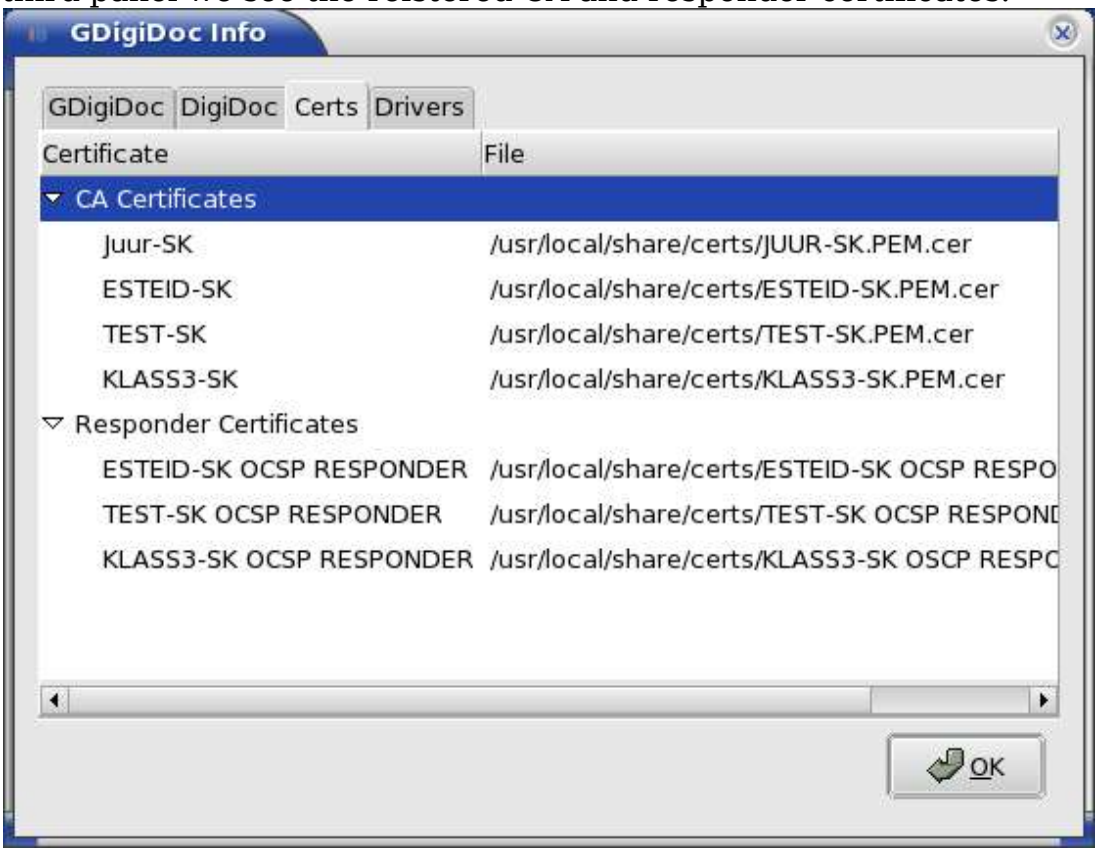
The menuitem Help -> Info opens the “About gdigidoc” dialogbox that displays copyright and other info.



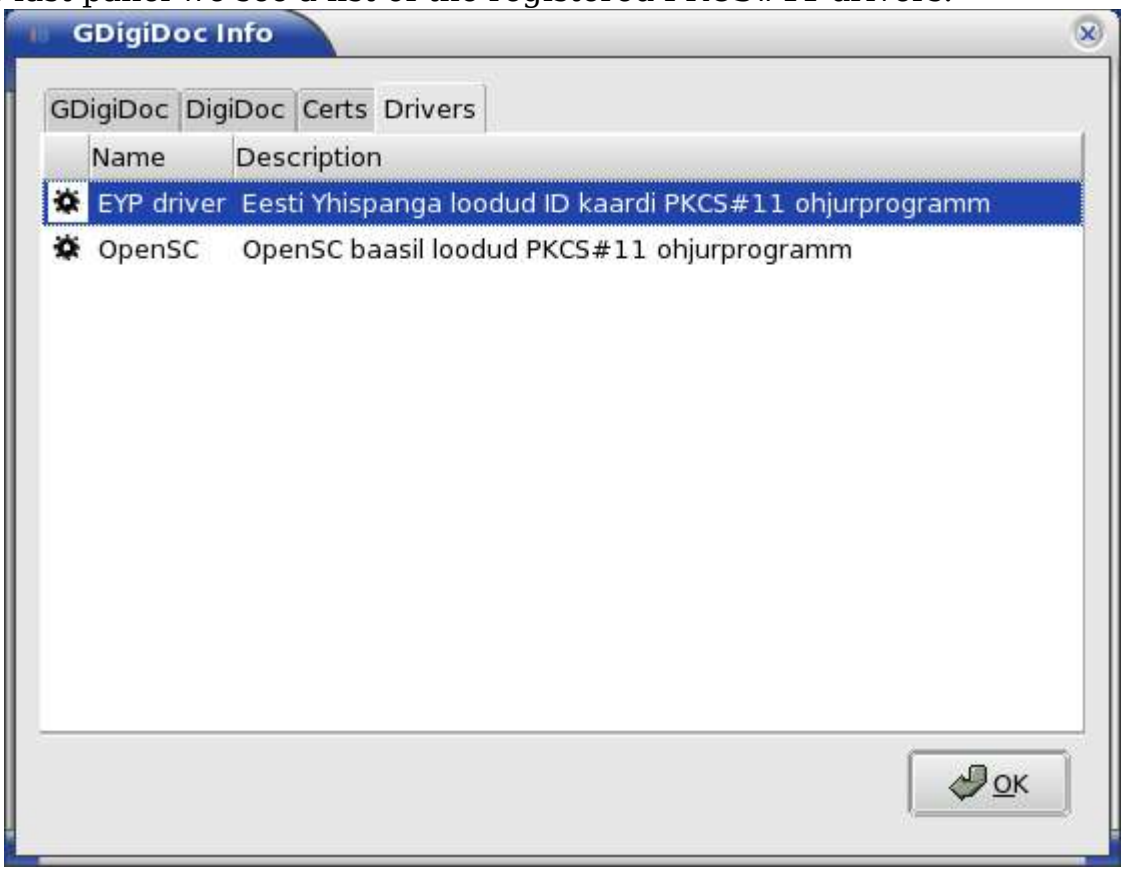
This dialog has four panels. The second panel displays information about the libdigidoc version used, supported file formats and configuration file settings.



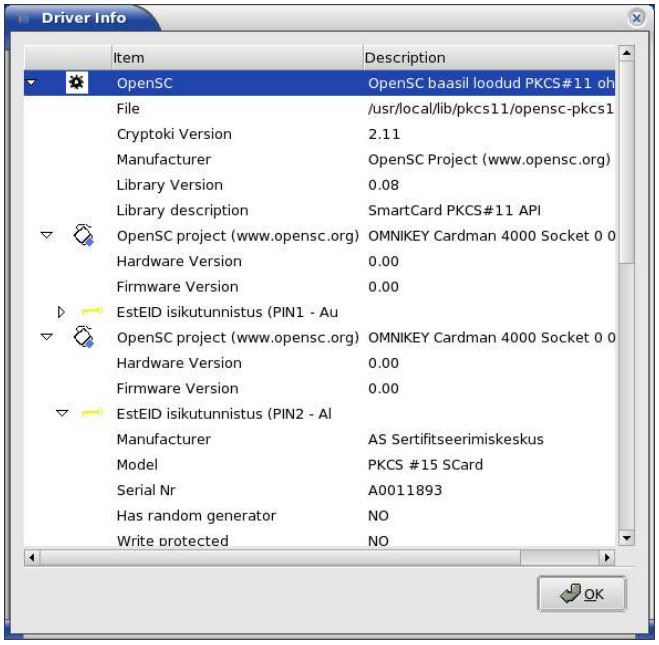
On the third panel we see the reistered CA and responder certificates.



On the last panel we see a list of the registered PKCS#11 drivers.



Here we see the driver provided by OpenSC library and another driver develeoped by Eesti Ühispank. If you right-click on one of the drivers in the list and select "Driver Info" the a dislogbox will appear and display a list of cardreaders and slots found using this driver. This is a good way to test if the program can access the smartcard.



References

- XML-DSIG – http://www.w3.org/Signature
- XAdES – http://www.w3.org/TR/XAdES
- OpenXAdES – http://www.openxades.org