Tcpconns
Key Store Manager GUI

User's Manual

Table of Contents

About

Welcome. Thank you for choosing tcpconns Keystore Manager application for all of your X.509 certificate needs.

With the tcpconns keystore manager you can open and save to different kinds of keystore types, including pkcs#12 (both old and new type), jceks, java keystores, uber formatted keystores, bks keystores, and OpenSSL's .pem formatted keystores. With the tcpconns keystore manager you can also act as your own certificate authority; you can generate self-signed root certificate authority keysets and you can approve certification requests. Of course, the tcpconns keystore manager also allows you and others to generate certification requests.

Note: the preferred type asymmetric key algorithm (for now) is RSA at a size of 4096 bits (i.e. > 128 bits symmetric equivalent); this type of key and key length value are considered to be secure for many years to come.

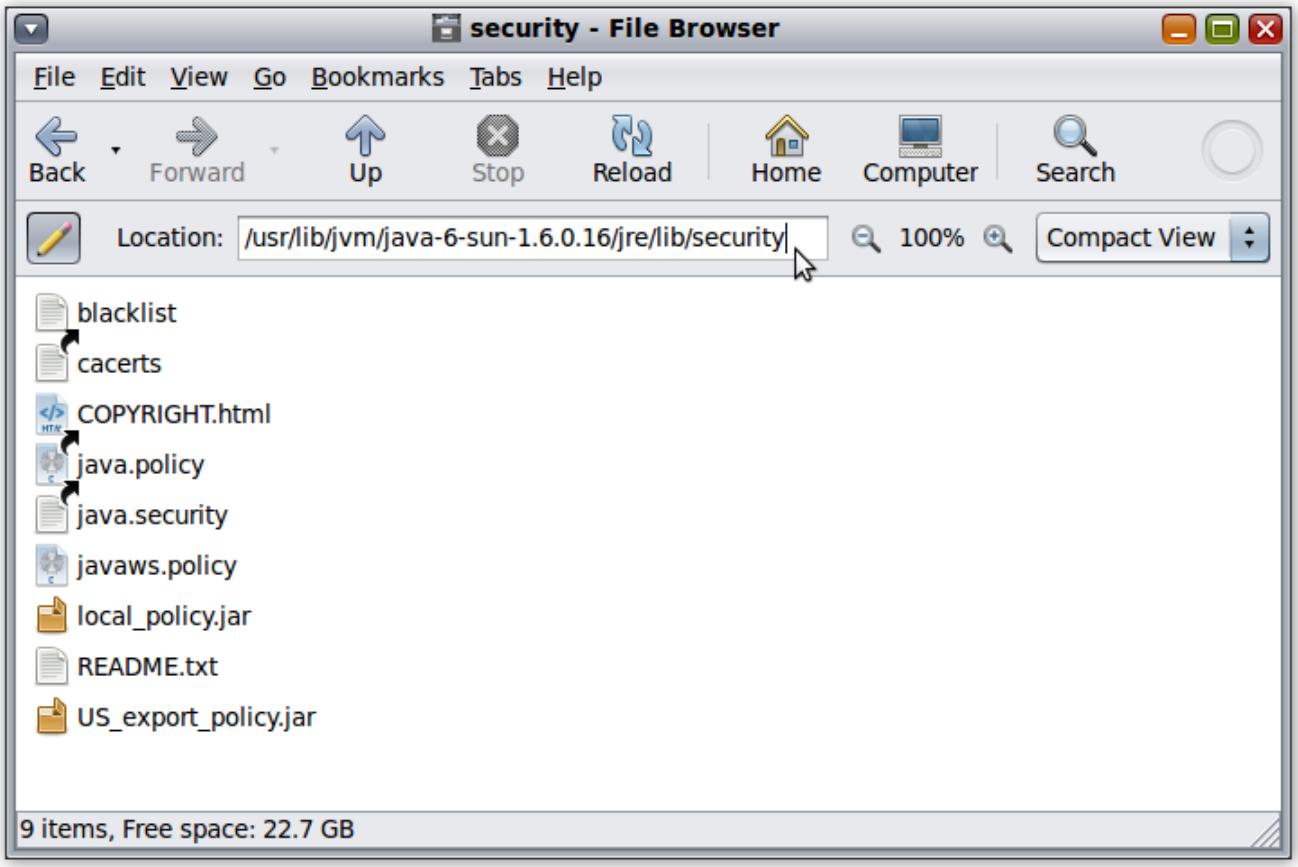
System requirements

Java 1.5/5 or higher

To be able to run the keystore manager application you need to have JAVA installed. Either the JRE (version 5 or higher), or the JDK (version 1.5 or higher). The SUN JRE can be downloaded **for free** on sun's website: http://java.sun.com/javase/downloads/index.jsp. JAVA JREs (JRE = java runtime environment) and JAVA JDKs (JDK = java development kit) are available for every major platform, including **Windows (98, XP, Vista+)**, **Macintosh**, **Linux**, **Solaris**/**OpenSolaris**, and **FreeBSD**/**PCBSD** (the latter can be found on http://www.freebsdfoundation.org/downloads/java.shtml, http://www.pbidir.com/bt/pbi/3/java_jdk). On Ubuntu e.g., the SUN JRE/JDK is available through the Synaptic Package Manager as well as opposed to a download from the sun website.

Unlimited Strength Jurisdiction Policy Files

To be able to use the keystore manager application, you will need to install the Java Cryptography Extension (JCE) **Unlimited Strength Jurisdiction Policy Files**, which can be downloaded from sun's website at http://java.sun.com/javase/downloads/index.jsp. Actract the archive/.zip-file you've downloaded, and place its contents in the /lib/security (\lib\security) subfolder of the JRE folder on your machine, accepting all overwrites. For this you probably need administrator priveliges ("sudo/su") on your machine.



Legal

THIS SOFTWARE IS RELEASED UNDER THE FOLLOWING BINDING LEGAL AGREEMENT (BSD-LICENSE).

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of tcpconns nor the names of its copyright holders or its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT

LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

IF YOU DO NOT AGREE TO THE ABOVE TERMS AND CONDITIONS THEN DO *NOT* USE THIS SOFTWARE AND *DO* DESTROY ALL COPIES OF THIS SOFTWARE THAT YOU HAVE.

If you make changes to the code, it is appreciated that you send your modifications to tcpconns@jhhcvandermeijs.nl so that there will be only one canonical version of the tcpconns library.

> TCPCONNS' CURRENT CODE IS NOT KNOWN TO VIOLATE ANY COPYRIGHTS OR INFRINGE ON ANY PATENTS. MOREOVER, IT ADHERES TO GENERAL PROGRAMMING RULES AND PRACTICES AND IS OF SUCH A SIMPLISTIC NATURE THAT A SIMILAR PACKAGE COULD BE CONSTRUCTED BY ANYONE.
>
> **IMPORTANT**: IF YOU FEEL THAT A PIECE OF CODE IN TCPCONNS VIOLATES A COPYRIGHT OR INFRINGES ON A PATENT, PLEASE DO INFORM ONE OF THE AUTHORS. DO ALWAYS STATE EXACTLY WHAT CODE IS NOT ALLOWED UNDER EXACTLY WHAT RULING, COPYRIGHT, OR PATENT; AND PLEASE DO ELABORATE ON EXACTLY HOW THE CODE VIOLATES THE SPECIFIC COPYRIGHT OR INFRINGES ON THE SPECIFIC PATENT.

N.B. Tcpconns uses the jars from the www.bouncycastle.org project.

N.B. Tcpconns includes a method Utils.getJarName which contains code originally written by "vafarmboy" (assumed to be public domain).

N.B. Tcpconns includes a method Utils.getNetworkIp which contains code originally written by Marcello de Sales (assumed to be public domain).

N.B. FTPServer includes code from the danoFTP project (http://sourceforge.net/projects/danoftp), which comes under a BSD license.

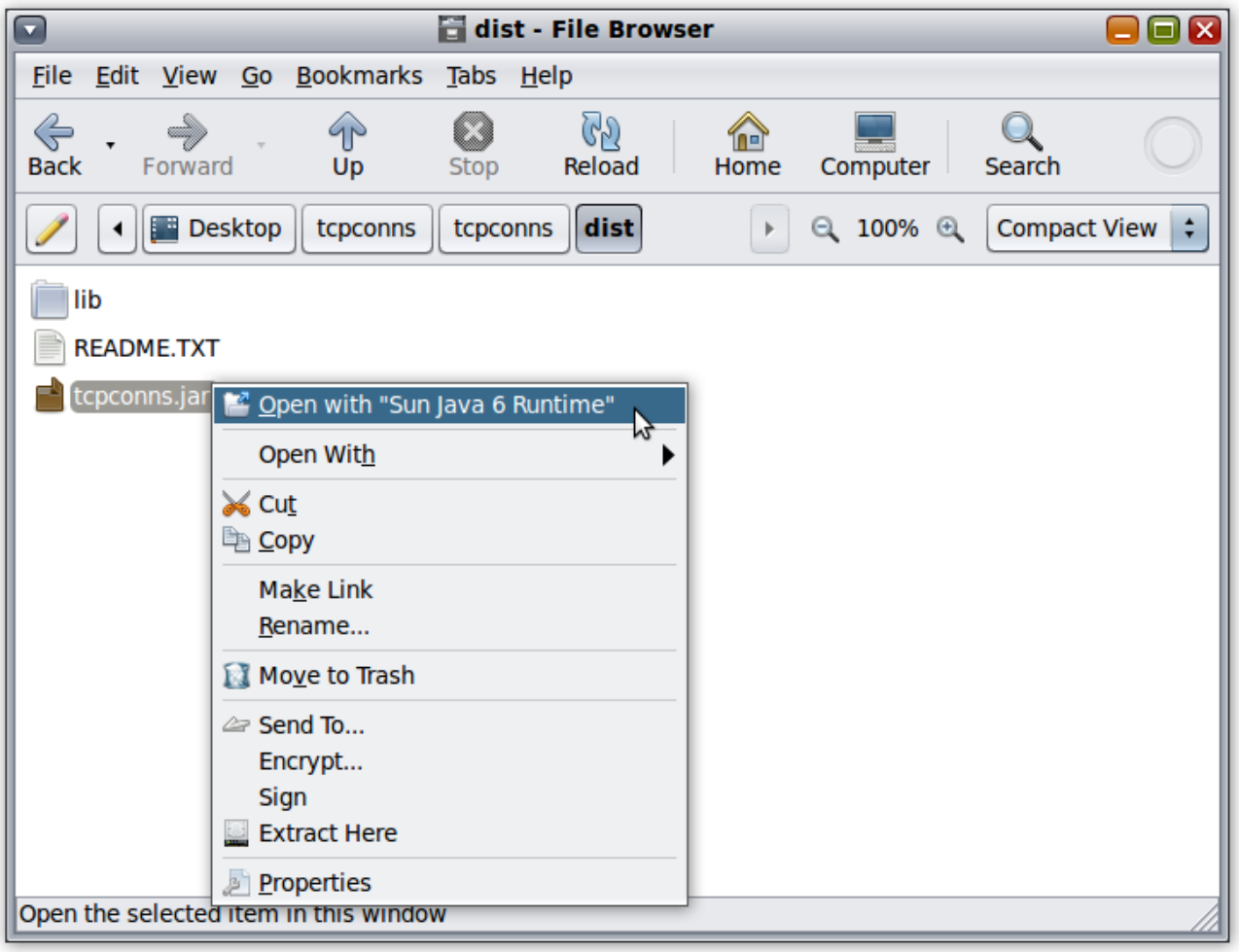
N.B. Tcpconns includes a class Utils.Base64 which is Copyright (c) 2004, Mikael Grev, MiG InfoCom AB (BSD license).

Tcpconns is Open Source Initiative approved Open Source Software. Open Source Initiative Approved is a trademark of the Open Source Initiative. http://www.opensource.org/.

Launching the key store management application

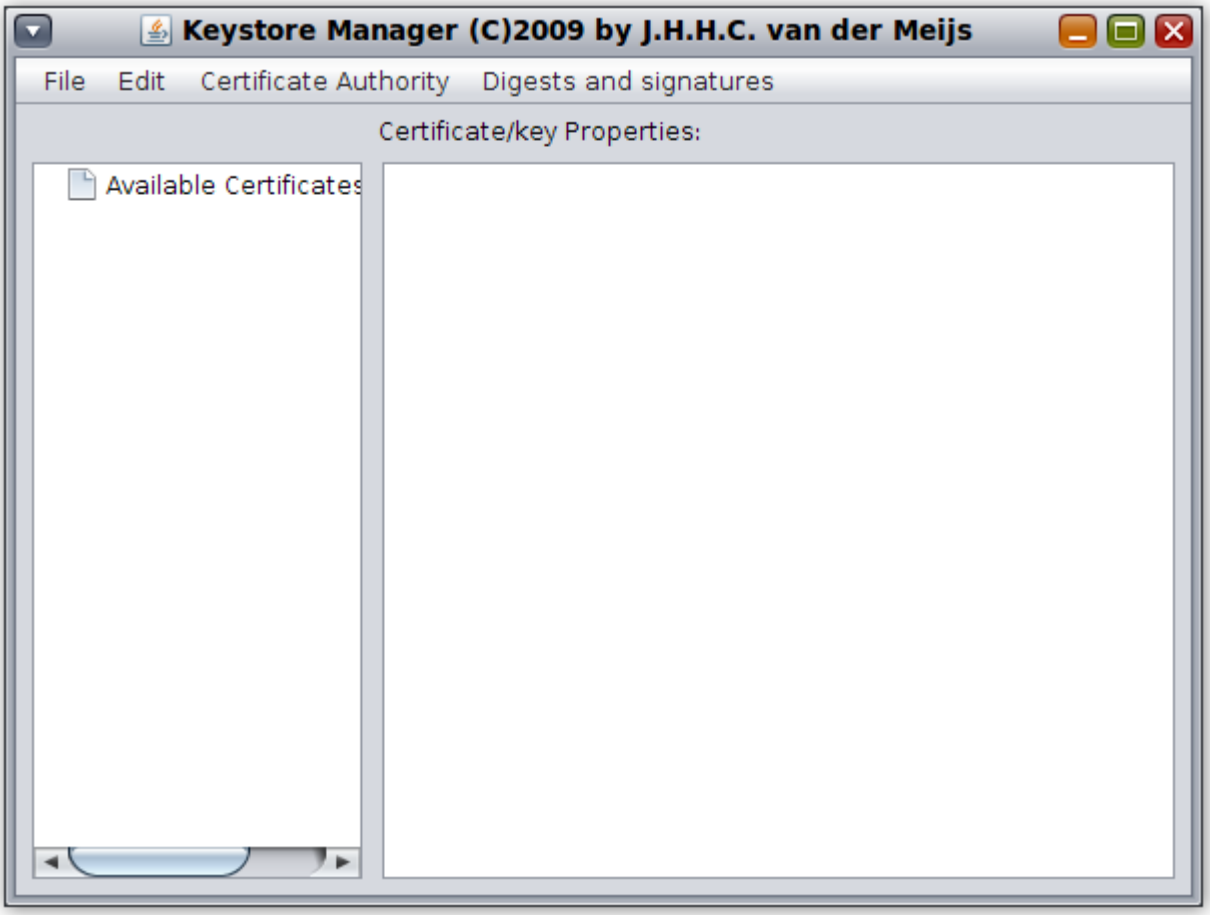
The key store manager GUI is started like any other java application. Locate the tcpconns.jar. Depending on your platform and your user settings, do one of the following:

- double click on the tcpconns.jar. This will either start the program or will open an archive manager tool (since a .jar is like a .zip). Altenatively right-click the tcpconns.jar and select “open with java”.

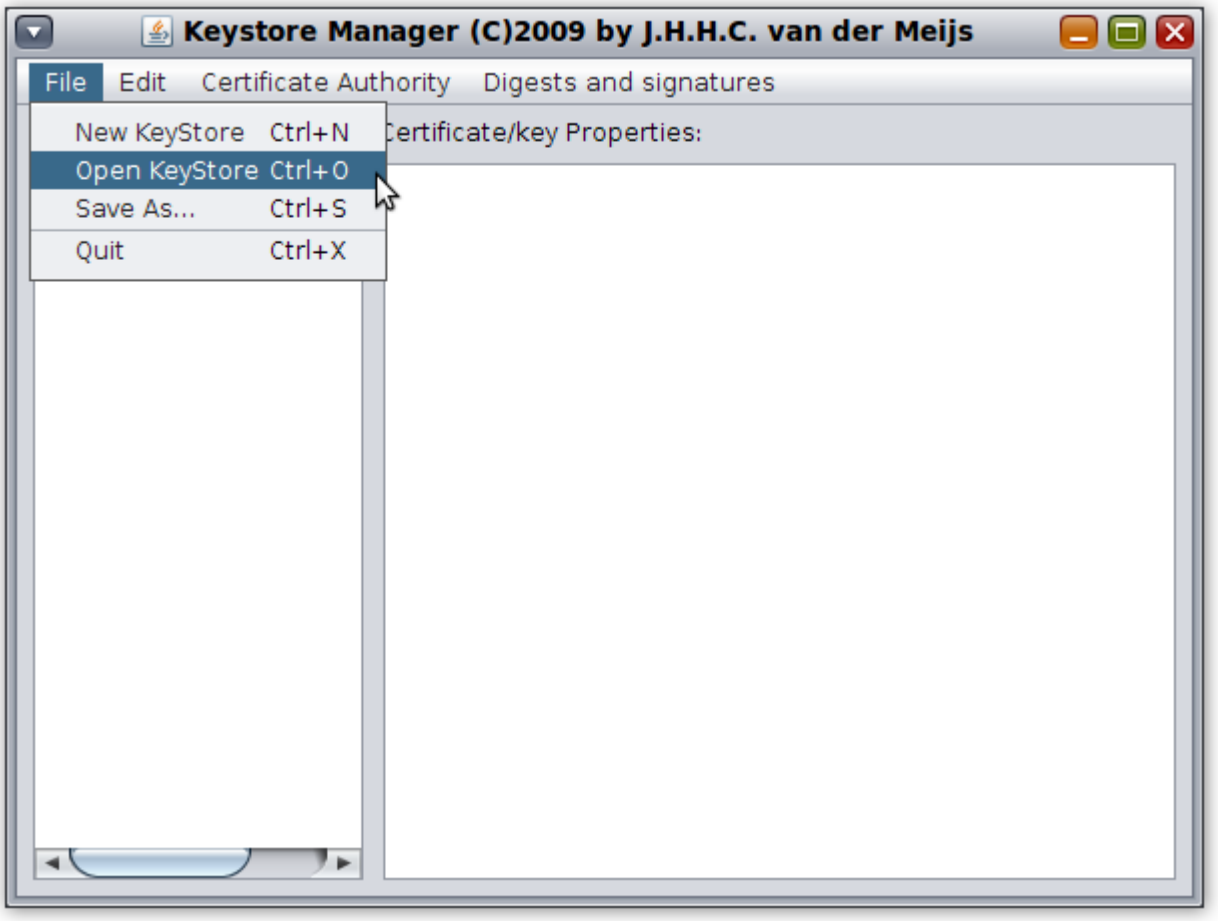


- open a terminal/dosbox and browse to the folder that contains tcpconns.jar. Then type “java -jar tcpconns.jar” and hit the [ENTER] button on your keyboard. This should start the program.
- create a link that does the same as the option directly above this one.

Once the KeyStore Manager GUI is running, it will display a window that looks like:

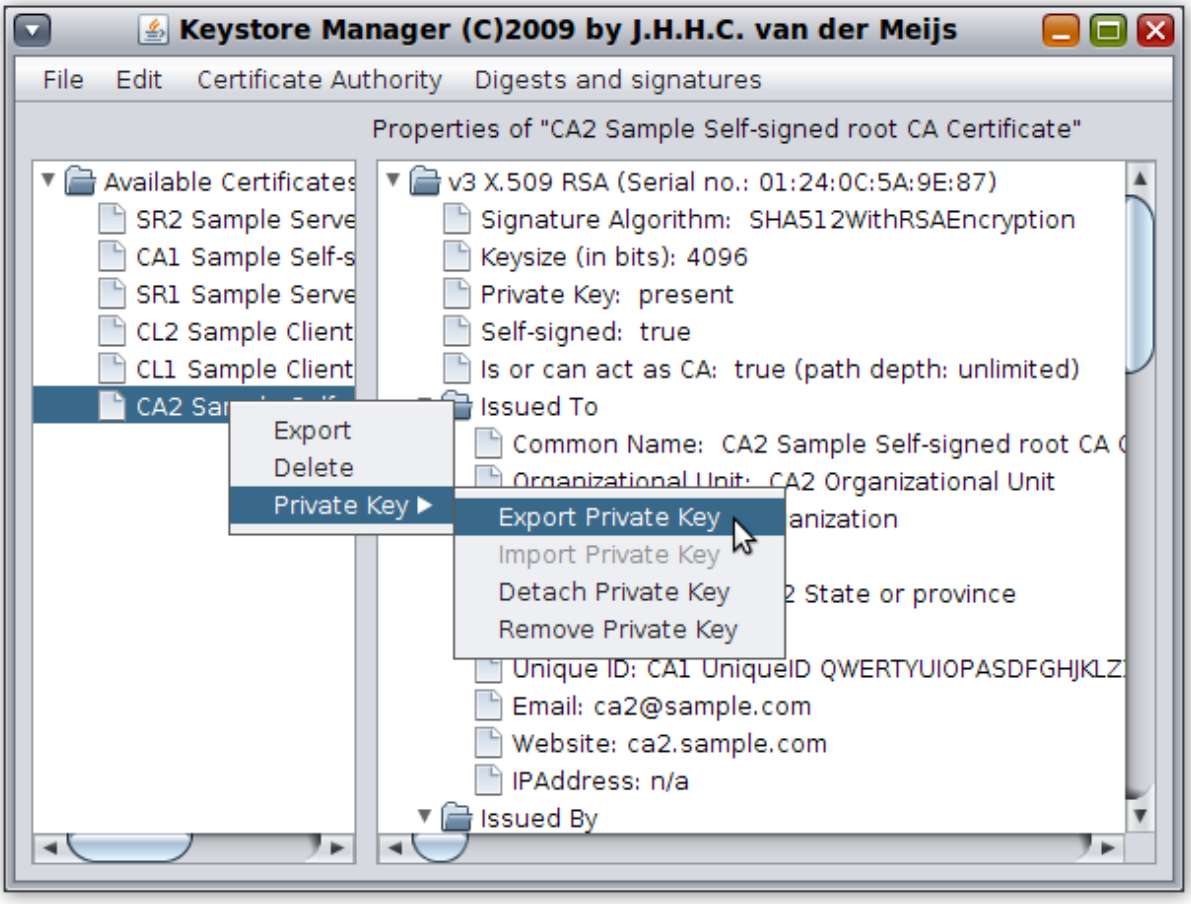


Basic keystore operations



1. **New keystore**: this will clear all contents from the keystore that is in memory. This will not affect any keystores on disk. N.B. to make changes permanent you will have to “Save As...” after you’ve performed the desired operations.
2. **Open keystore**: this will present an open dialog (browse to the desired keystore and approve it by selecting it and by pressing the open button) which is followed by a “enter passphrase” dialog. The keystore manager application is able to open keystores of the following types: pkcs#12 (both old and new type), java key stores, jceks, bks, uber type, and openSSL’s .pem type keystores. The keystore manager application can save keystores in these formats too.
    a. once a keystore has opened successfully, a list is displayed in the tree on the left. This includes all available X.509 entities with or without private keys.
    b. selecting an item (e.g. by left-clicking on the item) will allow inspection of the X.509 entity; the properties of which will be displayed in the tree on the right.
    c. opening a **context menu** on an item (e.g. by right-clicking on the item) will allow you to perform some additional operations:
        i. **export**: this allows you to export the X.509 entity, with or without private key, in various formats.

ii. **delete**: this removes the X.509 entity from the keystore that is in memory. This will not affect any keystores on disk. N.B. to make changes permanent you will have to “Save As...” after you’ve performed the desired operations.

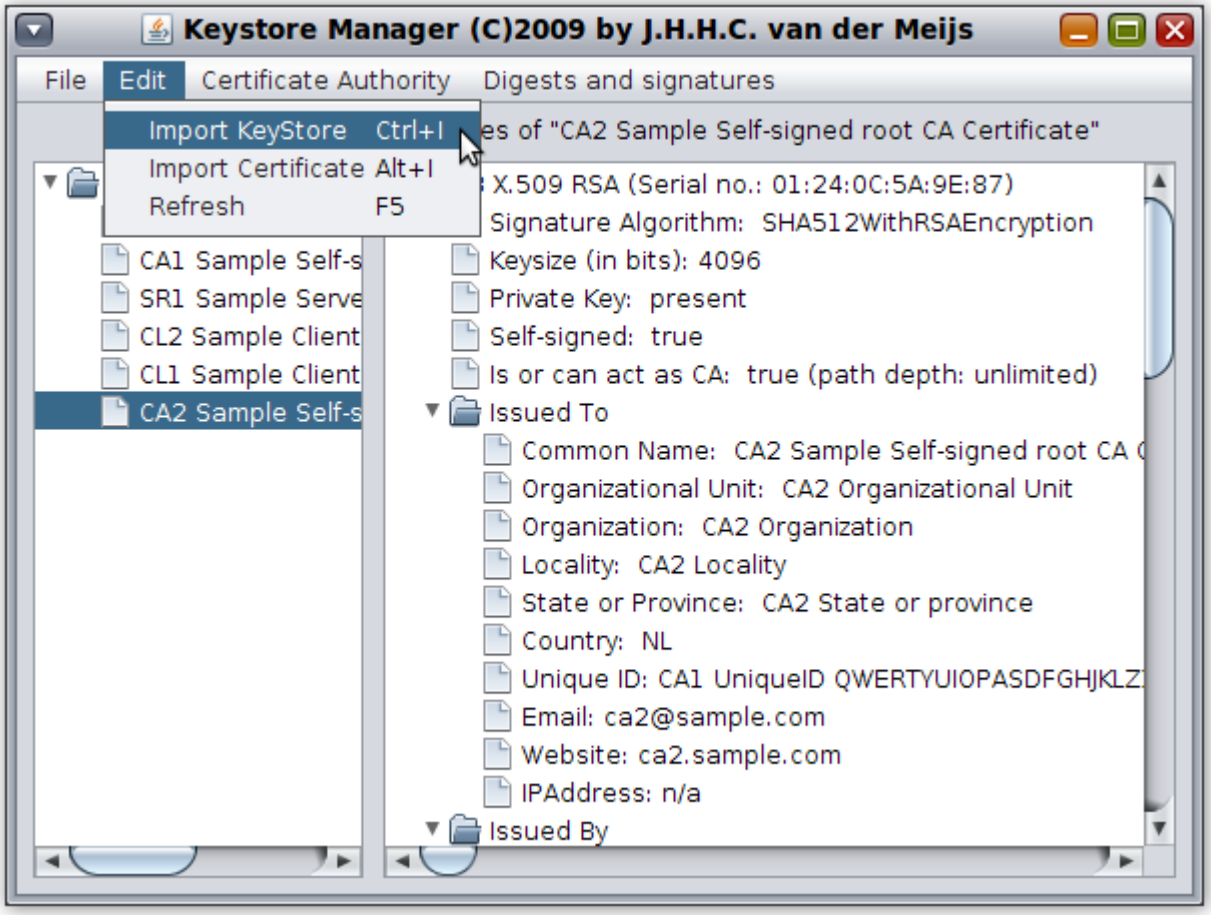


iii. private key

1. **export private key**: allows you to store the private key to disk (.pem formatted)
2. **import private key**: allows you to import a private key from disk (.pem formatted). This functionality is helpful if you have received your X.509 certificate from a certificate authority, and you are joining the private key with the X.509, so that you can store the combination e.g. as a pkcs#12 file or a .pem file. N.B. when you generate a certificate request with the keystore manager application, you must store both the request (.request) and key (.key) to disk. Once you have delivered your request to a certificate authority, the certificate authority can either approve the request by signing it (i.e. generating a X.509 certificate for you), or decline the request. Once approved, you will obtain an X.509 certificate from the certificate authority.
3. **detach private key**: the combination of export private key and remove private key
4. **remove private key**: this allows you to remove a private key from an X.509 entity in the in memory keystore. This will not affect any keystores on disk. N.B. to make changes permanent you will have to “Save As...” after you’ve performed the desired operations. P.S. please make sure that you have a copy of the private key left, stored somewhere safe; if this is your only copy of the private key, then it is needless to say

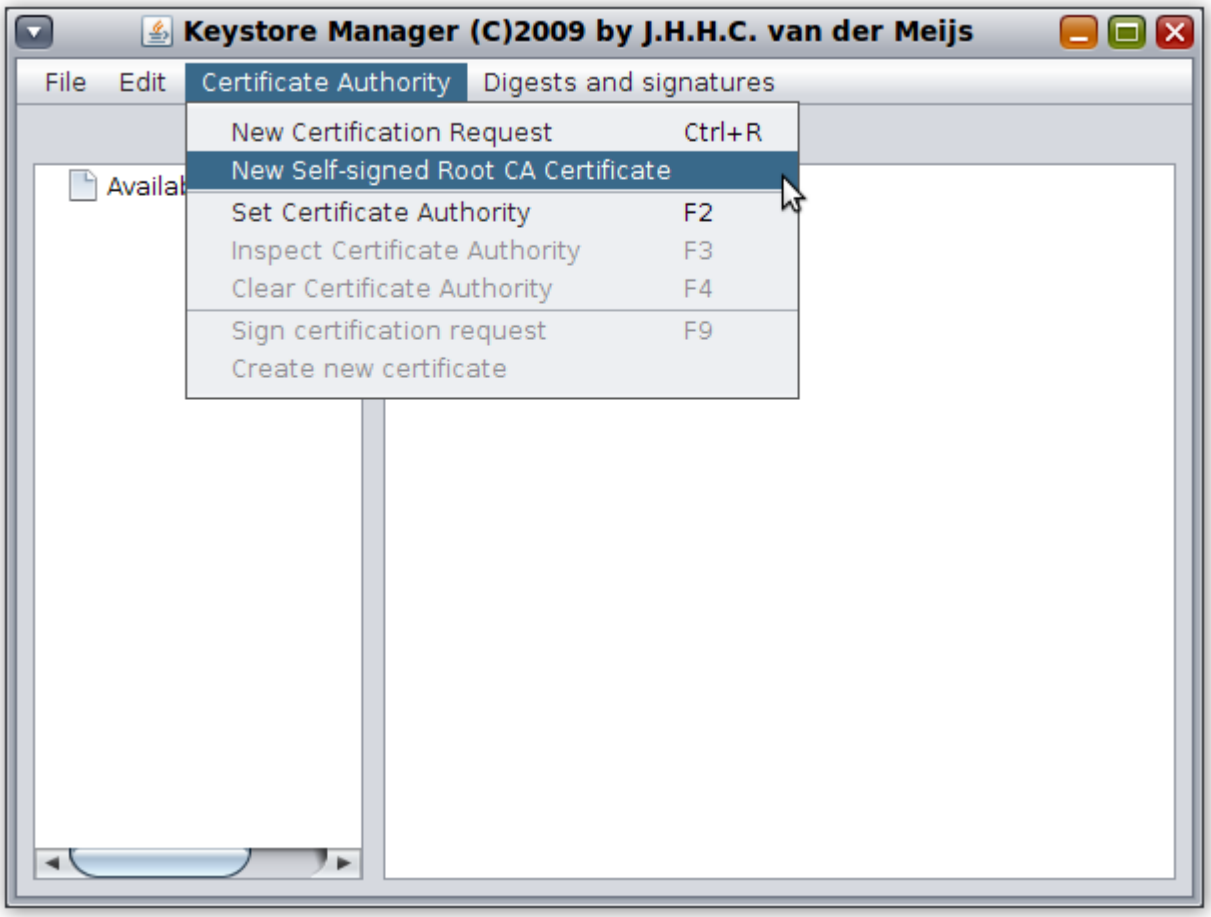
that you cannot use the private key any more once you remove it, and have overwritten the original keystore that contained the private key.

3. **Save As...** will allow you to save the in memory keystore to various types to disk. The keystore manager application is able to save keystores of the following types: pkcs#12 (both old and new type), java key stores, jceks, bks, uber type, and openSSL's .pem type keystores. Saving to old type pkcs#12 keystores is not recommended (sometimes it takes ages, the result cannot be used by some browsers/applications, and it is less secure than the new version of pkcs#12). N.B. private keys, even when stored with serious encryption methods still need appropriate handling and need not be made public.
4. **Quit**: quits the keystrore manager application.

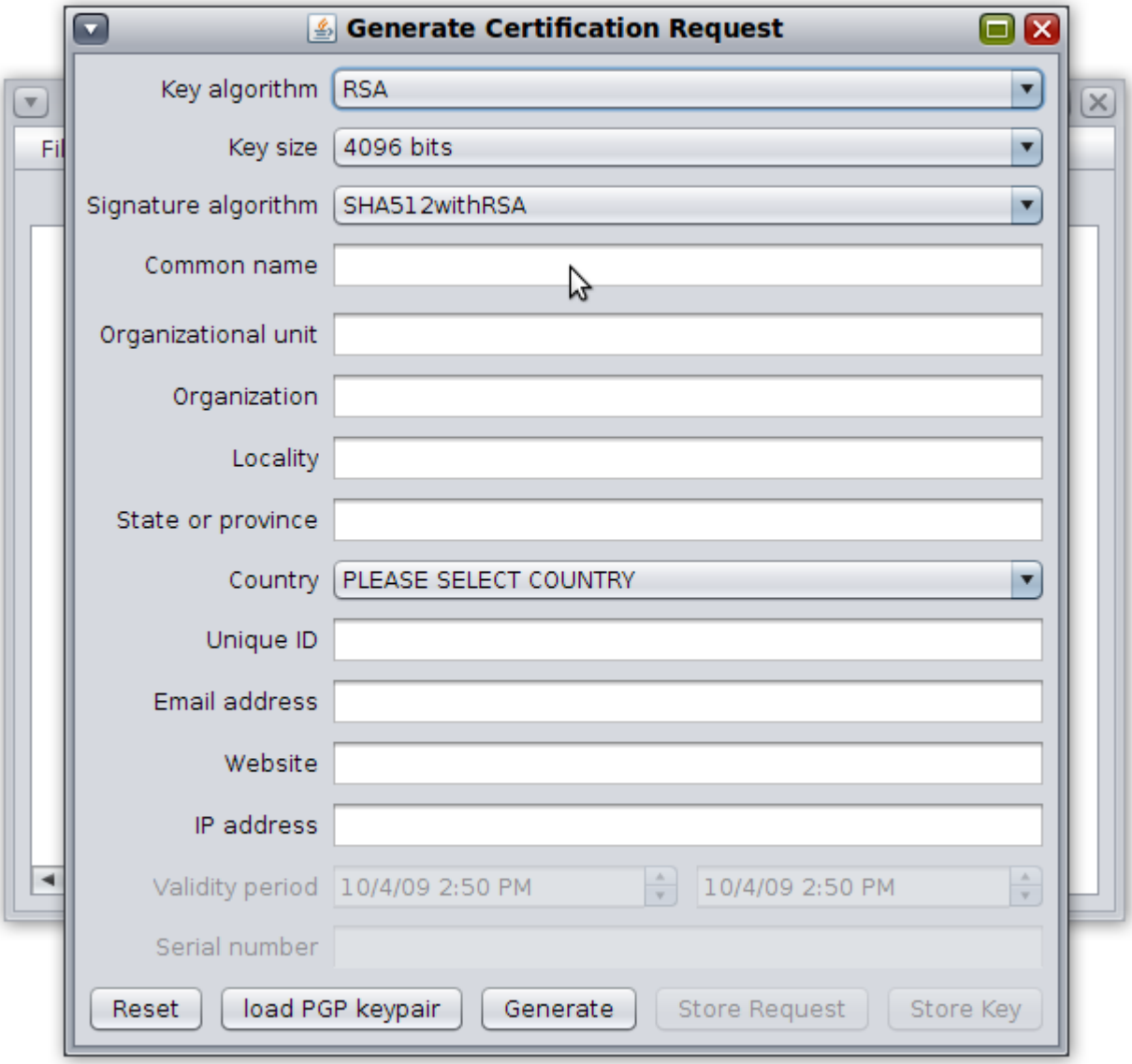


5. Edit:
   a. **Import Keystore**: allows you to add the contents of another keystore to this keystore. This will not affect any keystores on disk. N.B. to make changes permanent you will have to "Save As..." after you've performed the desired operations.
   b. **Import Certificate**: allows you to add a single X.509 certificate entity to the keystore (either .cer/.der-formatted or .crt/.pem-formatted). This will not affect any keystores on disk. N.B. to make changes permanent you will have to "Save As..." after you've performed the desired operations.
   c. **Refresh**: reiterates through the list of X.509 entities and reconstructs the tree on the left.

Certificate Authority Operations

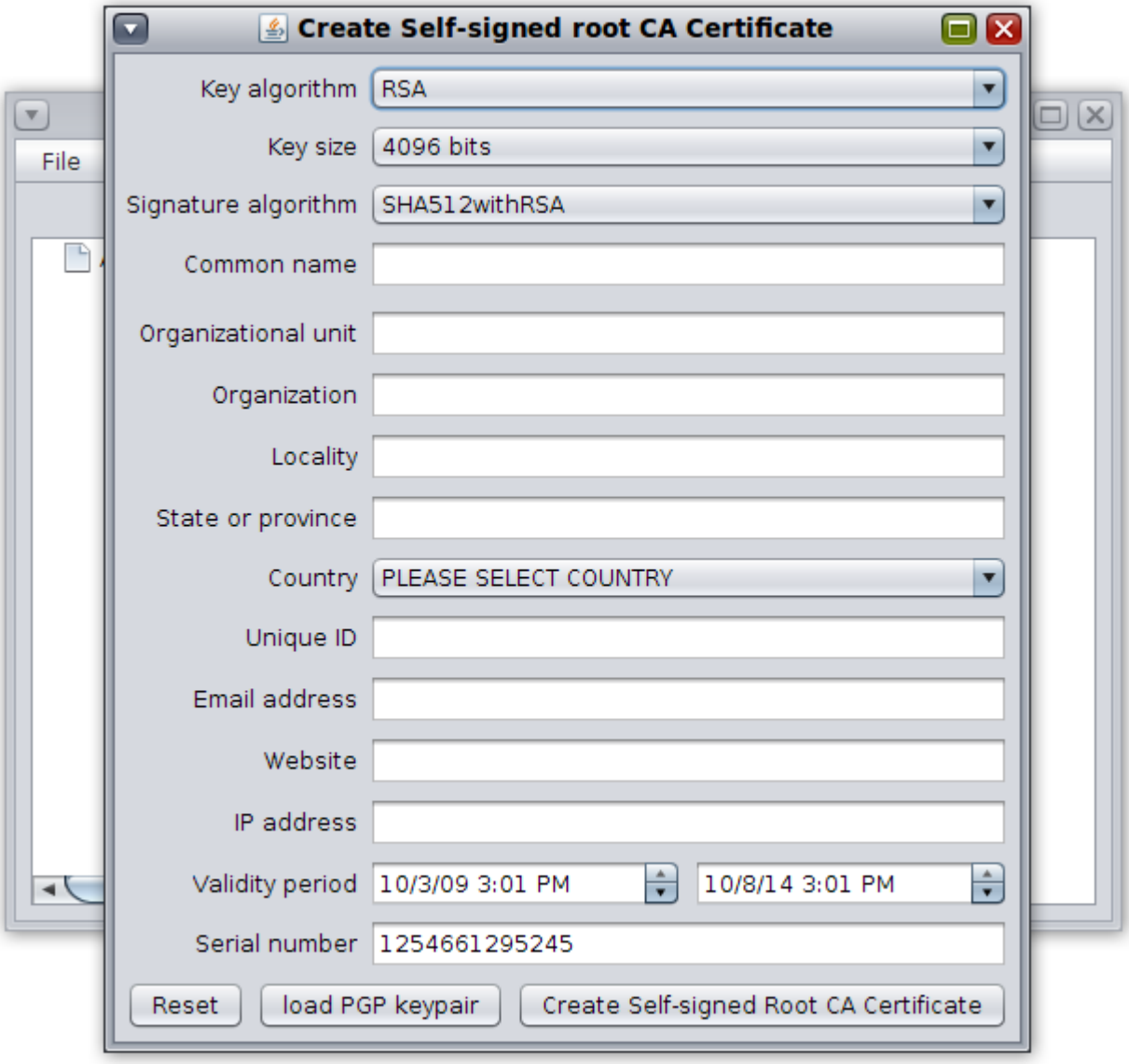


1. **New Certification Request**. Leave the settings for key algorithm, key size, and signature algorithm untouched, unless you know what you are doing (current settings are safe and interoperable/portable between different types of applications). Fill in all that is known. If you have a PGP keypair that you wish to reuse, then the “load PGP keypair” button allows you to import the PGP master keypair. Pressing reset will reset the dialog. Pressing Generate will lead to the generation of a new certification request.

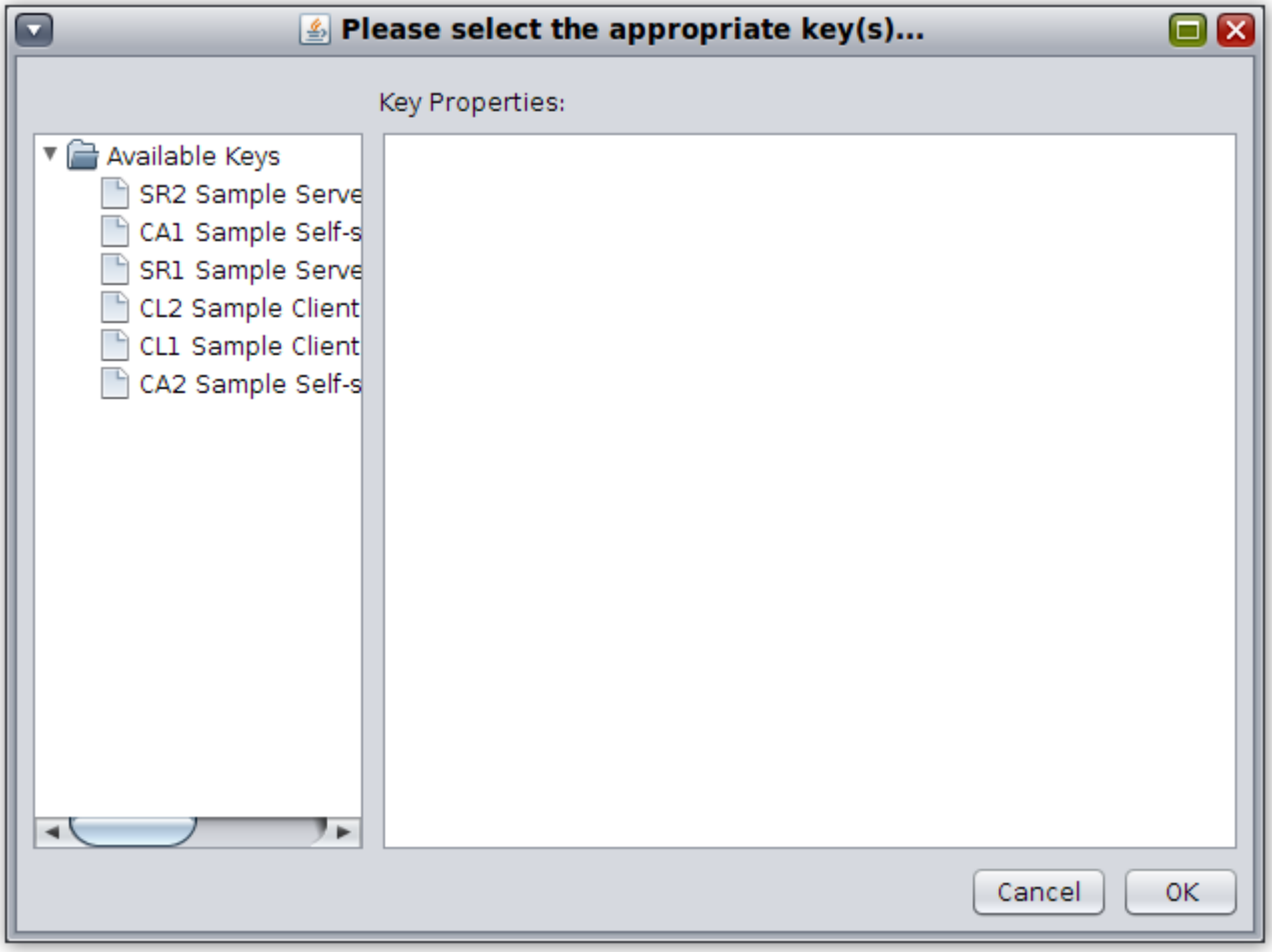


After a certification request is generated, please DO store the request (.request) and private key (.key) to disk **before doing anything else**.

2. **New Self-signed Root CA Certificate**. Similar to the “New Certification Request”-Dialog, except that now you can enter a validity period.

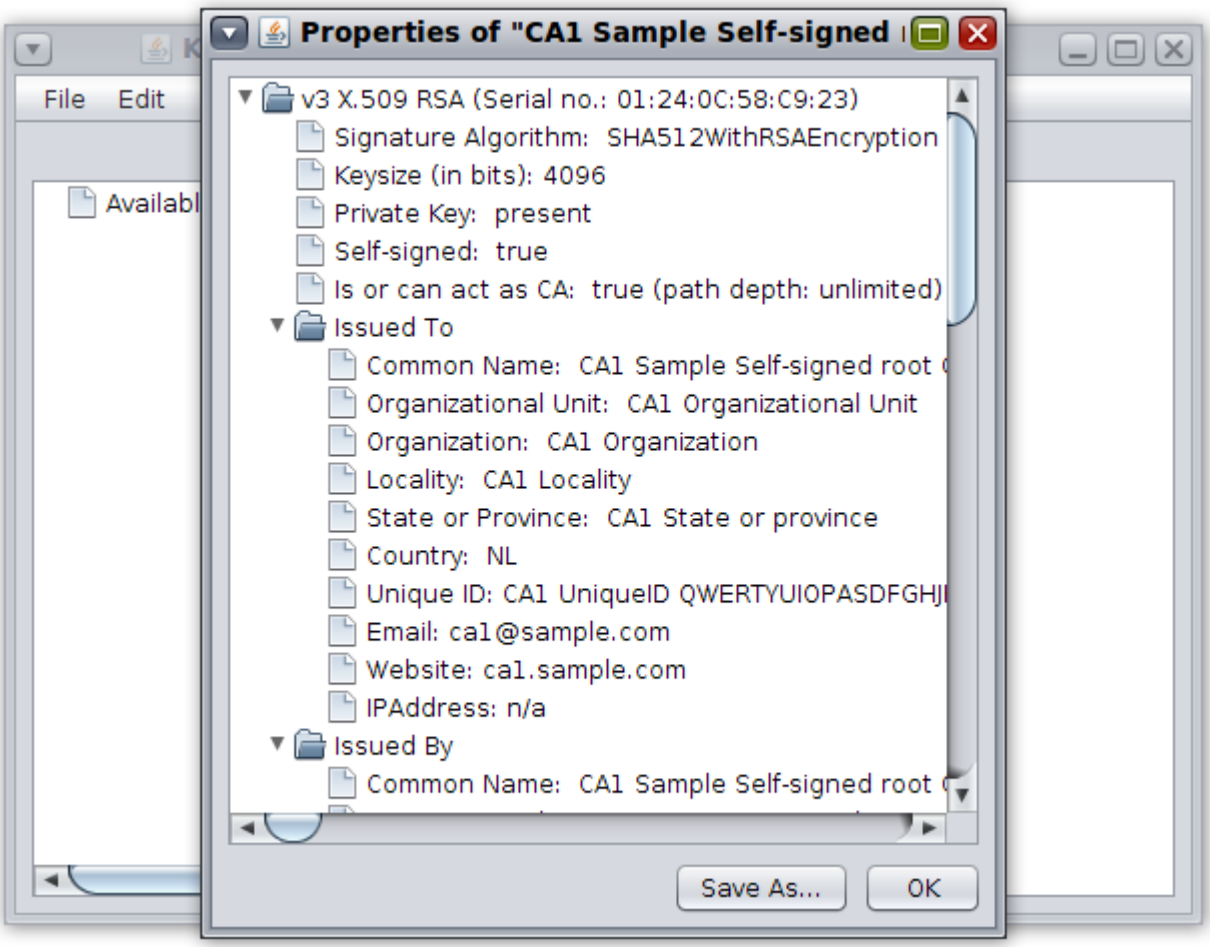


3. **Set Certificate Authority**. Allows you to select a certificate authority, which you then can use to approve certification requests. You will be presented with an open keystore dialog, a passphrase dialog, and subsequently with a “please select the appropriate key”-dialog.



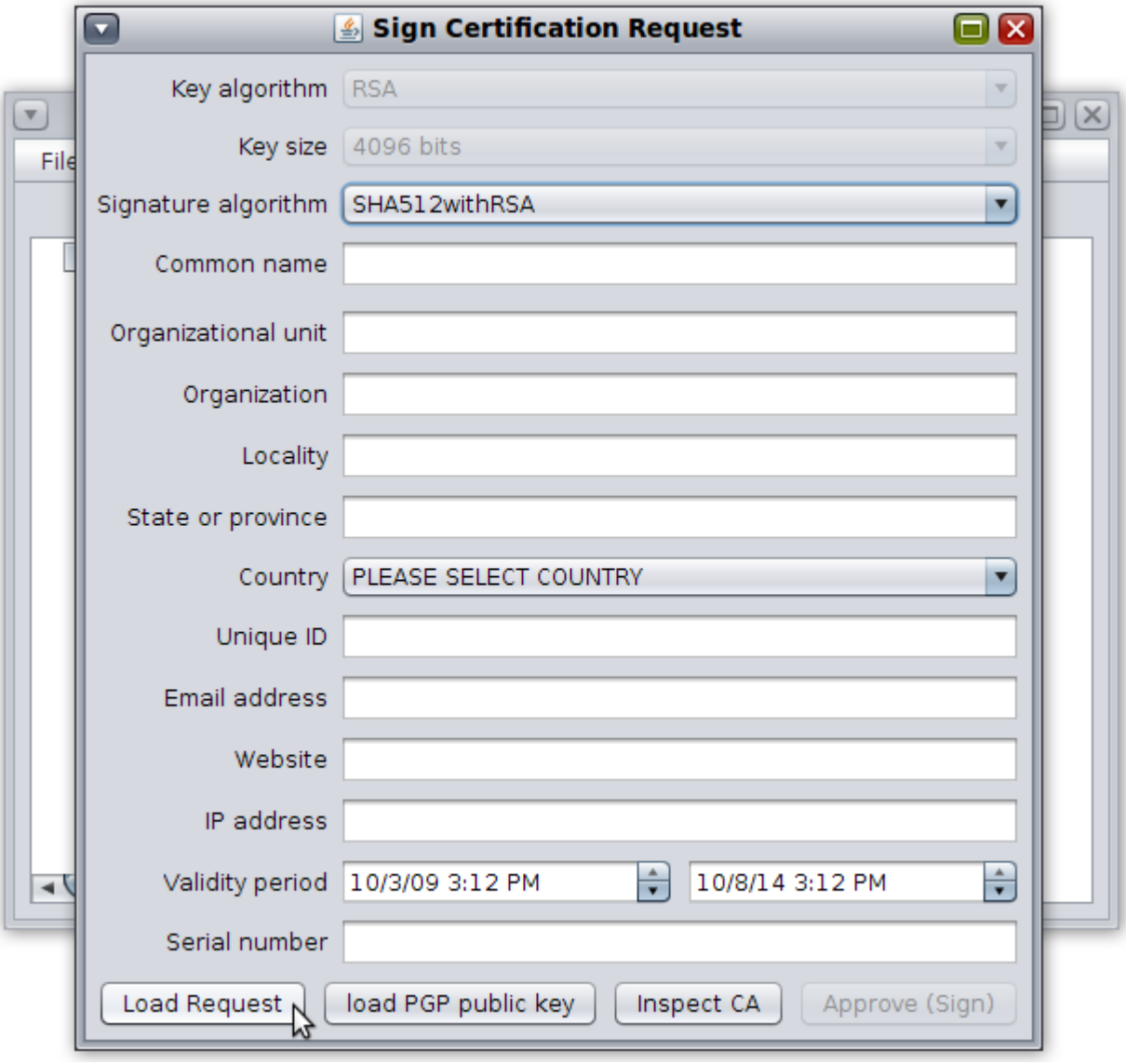
After selecting the desired Certificate Authority Keypair press the OK-button. A message dialog will confirm selection of the CA entity. N.B. in tcpconns’ keymanager application constraints set by others are ignored. **Any X.509 entity with private key can act as a CA**. Certificates generated and certification requests approved with the keymanager application will all be official certificate authorities with unlimited path lengths.

4. **Inspect Certificate Authority**. Once a certificate authority is set, you can inspect the certificate at any time.



5. **Clear Certificate Authority**. Makes the keymanager application stop acting as a CA.

6. **Sign Certification Request**.

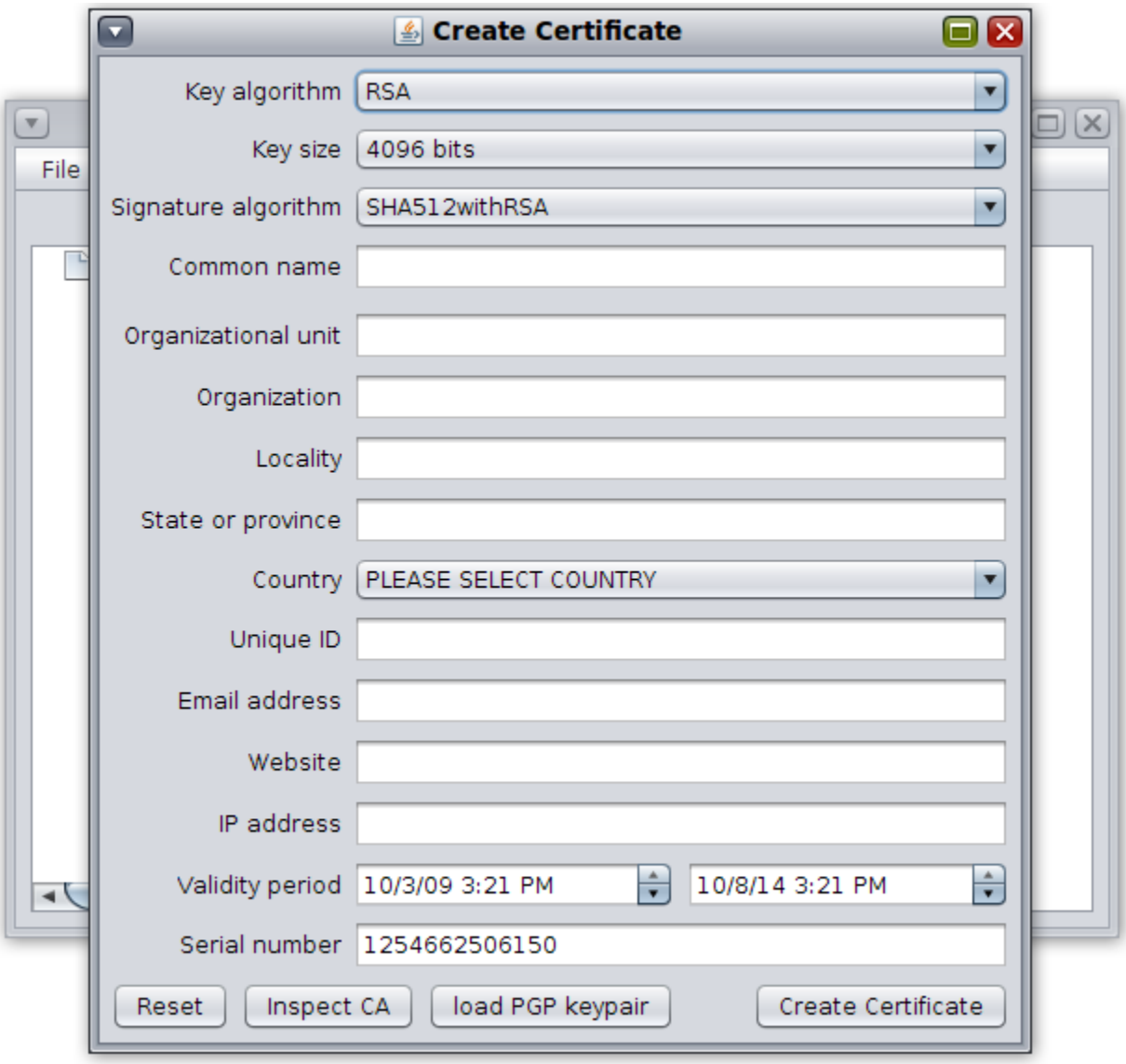


First, press the load request-button. This checks the integrity of the request and fills in the form.

The “load PGP public key”-button serves only one purpose: to extract the validity period from the given pgp public key. Note: nothing else is taken from the PGP public key pair. When should you need this functionality? When you’ve used a PGP keypair to construct a certification request, and you wish that the X.509 will generate the same KeyID as your original PGP keypair. This is necessary for creating and verifying OpenPGP signatures made with X.509 entities: in this way GNUPG and PGP will recognize signatures made by your newly generated X.509 keypair.

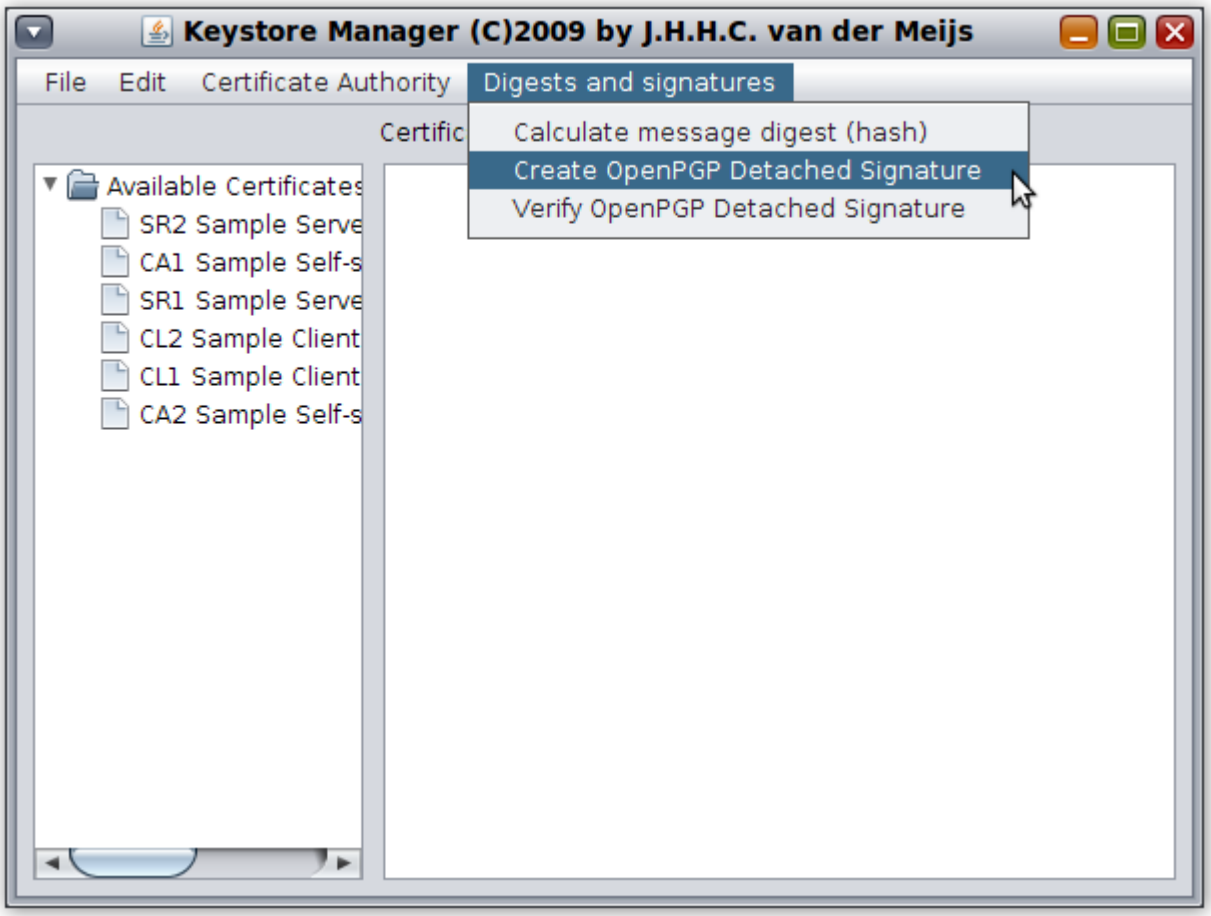
Once you have verified all data and found it to be correct (be very meticulous about this), then hit the approve (sign)-button. **N.B. You should only sign certification requests of which you know the origin and know for certain that the identity information supplied with the request belongs to the person requesting the certificate to be signed.**

7. **Create New Certificate**. Allows you to create a new certificate with the set CA as signing party. Under the hood a certification request is generated and approved.



Miscellaneous

Digests and Signatures



1. **Calculate message digest**. This will open a dialog in which you can select a file or type a text, and the generate various message digests with it. They are placed in the text area. If it is a large file, then it may take some time, and the dialog may be unresponsive.
2. **Create OpenPGP Detached Signature**. First opens a dialog in which you can select a X.509 entity that has a private key attached. Prior to this you must have opened a keystore! Then an open dialog will appear: select the file for which you wish to create a signature. Then a save dialog will appear: enter the file to which the signature must be saved. Signatures are saved in ASCII format.
3. **Verify OpenPGP Detached Signature**. An open dialog is presented in which you must choose the file from which you wish to verify a signature. Another open dialog is presented, please select the signature file. The keystore manager application will search among the X.509 entities in the keystore that is in memory (i.e. the keystore that you opened before starting the verification process), and if the signing key is present, it will verify the signature. The **public** key of the signing keypair must be present to be able to verify the signature. For now it only verifies that the Message Digest the signature contains is signed by the signing keypair and that it corresponds to the Message Digest that was regenerated for the specific file.