PKI-Linking-Report: Connecting Public-Key-Infrastructures

Version 1.0.2

Olaf Gellert, April 2005

This document was created in a joined effort of SURFnet and DFN-CERT Services GmbH. The work was carried out under subcontract by PRESECURE Consulting GmbH. The work was conducted by Olaf Gellert.

This report is furnished on an "AS-IS" basis. The parties engaged in this project - SURFnet b.v., DFN-CERT Services GmbH, PRESECURE Consulting GmbH - make no warranties of any kind, either expressed or implied, as to any matter including, but not limited to, warranty of fitness for purpose or merchantability, exclusivity, or results obtained from use of the material. The parties to not make any warranty of any kind with respect to freedom from patent, trademark, or copyright infringement.

Use of any trademarks in this report is not intended in any way to infringe upon the rights of the trademark holder.

Internal and external use: You are free to copy, distribute, and display the work, and to make derivative works under the following conditions:

By Attribution: You must give the original parties credit.
Noncommercial: You may not use this work for commercial purposes.
Share Alike: If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

See the Creative Commons Attribution-NonCommercial-ShareAlike Licence [CrCo2004] for further details.

Requests for permission to reproduce this document or prepare derivative works of this document for external and commercial use should be addressed to either one of the parties engaged in this project.
Abstract

After many years of puffery and big promises, PKI has reached a state of prosaic and down-to-earth use. The visions of global, unified PKIs did not come true, nevertheless PKI is in a state of simply being used and useful. Many organisations run their own PKI now, creating certificates predominantly for signed and encrypted email and for user authentication. On the one hand this leads to a growing amount of users that own certificates and know how to use them, which provides a basis for further development of PKI-enabled applications. On the other hand many separate small PKIs have been established that could be much more useful if they could be linked to each other. Linking these PKIs would enable secure information exchange between different institutions.

A common way to link different PKIs is the usage of cross certificates, where the certification authority (CA) of one PKI signs the certificate of another CA. This expands trust from the local PKI into the other. It also raises a few issues: Should every local client trust all certificates from the other certification authority? Email clients should accept signed emails from the other PKI as being trusted. But usually these certificates should not be valid to access the local institution's internal network. The evaluation of additional knowledge (e.g. in form of policies) is necessary to decide if a certificate is to be trusted. Another issue is that cross certificates have been a subject of many proposed solutions for building trusted relations between PKIs. But these solutions are highly dependent on the applications that must be able to cope with cross certificates. This report reviews the possible solutions for linking of PKIs and provides some first results of tests with actual applications and certificates.
Table of Contents

Abstract.......................................................................................................................................1
Introduction................................................................................................................................3
Methods of linking different PKIs..............................................................................................5
Establishing a Test Framework for PKI Linking Methods.......................................................10
  General Concept...................................................................................................................10
  Creating the Certificates......................................................................................................11
  Choosing the test applications.............................................................................................12
  Setting up the testbed...........................................................................................................12
Tests..........................................................................................................................................15
  Test Scenario 1: PKCS12 Root Certificate Distribution......................................................16
  Test Scenario 2: PKCS7 Root Certificate Distribution........................................................32
  Test Scenario 3: SuperCA PKI Federation............................................................................38
  Test Scenario 4: Cross Certificate, Root signs Root............................................................47
  Test Scenario 5: Cross Certificate, Root signs SubCA.........................................................56
  Test Scenario 6: Cross Certificate, SubCA signs SubCA.....................................................61
  Test Scenario 7: Root-Certificate-Exchange with Cross-Certificates..................................66
Results......................................................................................................................................76
Appendix A: Certificate Profiles..............................................................................................82
Appendix B: Test Framework Scripts.......................................................................................84
  Introduction to the modules and scripts..............................................................................84
  Setting up the test PKIs........................................................................................................87
Bibliography.............................................................................................................................88
Introduction

Today public key infrastructures are established in many organizations to enable secure communication and authenticated access to internal services. While PKIs have become a part of the day-by-day business of many organizations, secure communication with external communication partners is still not the rule. While most of the common applications are X.509-enabled, further features supporting the widespread use of certificates like automatic key retrieval are still missing. Especially the problem of initial trust in other PKIs root certificates remains unsolved. X.509 PKIs use a strictly hierarchical model, so the certificate structure is always a tree. To validate a certificate, the path of issuing CAs is followed up the certificate tree while checking each certificates signature against the issuing CAs key. While in a PKI all but root CAs certificates are signed by the issuing CA, the root CA certificate can only be signed by its own key. Hence the root certificate cannot be validated automatically, the trust in the root CA key has to be given by some other means eg. by manual evaluation of its fingerprint. The root certificate is therefore the trust anchor of the complete PKI underneath. In a closed environment the authentic distribution of the root certificate may be easy, worldwide communication impose other problems. There is no root CA which is the single worldwide root for all existing PKIs (this would imply that the CA would be trusted by the whole world without exception). Instead we find islands of PKIs for different organizations and purposes which all have their own trust anchors. These must be distributed in a secure manner. Actually the trust anchors of external PKIs are either preinstalled in distributed software (a common practice in webbrowsers) or have to be installed manually. In the case of the common user the implications of PKI trust are often not understood, queries of the software concerning unknown certificates usually leave a feeling of uncertainty and discomfort with the inexperienced user.

So the linking of different PKIs remains a challenge in a world with a growing need for secure communication. Different solutions have been proposed in the past, from manual certificate installation to fully automated distribution of trust anchors. While some of these solutions are already part of PKI standards (for example [RFC 2510] makes broad use of cross certificates) it usually remains uncertain if the actual applications support these. This lack of knowledge obviously slows down the further process of PKIs growing together which in turn diminishes the usefulness of todays PKI applications.

Especially in the case of the European research networks many independent organizations are linked together. Different PKI solutions and environments are common while the need of sharing knowledge and resources is growing. To be able to advise the participating organizations and to provide the necessary knowhow, SURFnet and DFN-CERT Services GmbH conducted some tests to get a first insight into the actual options to link PKIs. A set of common usage and the according applications were chosen as test cases. As heterogenous environments are most common nowadays, different operating systems had to be tested. For each of these possibilities a range of solutions for the distribution of trust anchors was tested.
This report presents the choices of applications and linking methods to be tested. The complete test environment is described to provide a platform for further testing. The report contains detailed descriptions of the necessary work to establish trust anchors of new PKIs in the applications. To distribute this knowledge and to encourage further research in this field, this report is published under a Creative Commons License [CrCo2004] which allows sharing and further distribution of these results.

**Acknowledgements**

Thanks to Jan Meijer from SURFnet for his patience, his encouragement and many good ideas for this report. Thanks to SURFnet and DFN-CERT Services for providing the means to do this work. Thanks to the DFN-Verein (German Research Network) for the continuous support of my PKI work. I have to thank Klaus-Peter Kossakowski for allowing the necessary time for this effort. And finally thanks to all others who were suffering in any way from me being occupied with this work.

February 2005

Olaf Gellert
Methods of linking different PKIs

One of the problems in establishing a PKI is the distribution of the root certificate of the certification hierarchy. As described above, the root certificate is signed only with its own key, so there is no initial trust path to this key. In many client applications, some root certificates of commercial certification authorities are preinstalled to solve this problem. But this practice basically means that these CAs are trusted by default (which may not be desired by all users or for all use cases), while all other CAs are not trusted. Applications are often able to add trusted CAs to their certificate store. Sometimes for each CA specific trust levels or application purposes may be defined.

Thus a solution of the problem would be a method to integrate foreign root certificates in a trusted way. Publishing the root certificate and the according fingerprints on a web page is always a good idea to allow for at least a minimal validation of the certificate. Unfortunately today’s applications are not prepared to guide a user to such a webpage when receiving a new root certificate, so this always requires manual interaction by the user.

If the CA generates not only the certificates but also the keys for the users, a secure way of distributing the secret keys has to be established in any case. The root certificate of the PKI can be distributed in the same way as the keys, usually it is even provided in the same file. The standard message format for this is PKCS#12 [RSA 1999], which always contains an encrypted secret key and additionally some certificates. The password for the encrypted portion of the PKCS#12 file still has to be submitted in a secure way.

Using this method, an organisation running multiple PKIs can choose to distribute all the different root certificates together with the users PKCS#12 file to enable an easy use of certificates from all PKIs.

A general procedure to reduce the efforts of distributing a whole bunch of root certificates is to bundle the certificates in one file which is signed by a kind of Super CA. In this case the user has to validate only the certificate of the Super CA and can then import all the root certificates in the file. A standard file format to distribute certificates is PKCS#7 [RSA 1993]. The PKCS#7 "signed-data content type" supports signed application data, it can also be used to disseminate certificates by entering the certificates into the certificate set of the "signed-data" field. Actually it would be possible to enter root certificates into the contentInfo field, but the interpretation of the data is left to the applications. Most applications do not seem to support the signed dissemination of root certificates in PKCS#7 files (as well as OpenSSL does not appear to support the generation of these files).

Still PKCS#7 files can be used to distribute a set of certificates and were therefore included in the tests.

While the previous process uses the Super CA only to sign a file containing all the other CA certificates, a Super Root CA can be established issuing new certificates for the CAs. If the
attributes which are used to build a certificate chain (usually the attribute authorityKeyIdentifier) allow for some flexibility, the CA certificates can be reissued without harm to the certificates further down the chain. That way PKIs are linked together like one single PKI. Usually there are two ways helping the clients to validate the certificate chain. Either the authorityKeyIdentifier contains a hash derived from the public key of the signing CA or it contains the subject name of the CA two levels above and the serial number of the issuing CA certificate. The first way is very flexible, it allows exchanging the CA certificate for a new one, as long as the key of the CA stays the same. The second way is more static, an exchange certificate would have to use the same serial number, so the new CA certificate would be indistinguishable from the old one for most applications. This procedure would vulnerate common X.509 standards like PKIX (because a CA must not issue certificates with the same serial number).

All these methods are based on the different import mechanisms of trusted CA certificates into the applications and require that the user actively imports these certificates. This may not be a problem in large, centrally administrated organisations, in which the applications can be installed together with the necessary certificates. If linking between PKIs of different organisations is required (which is usually the case with email applications checking the validity of signed mails from other organisations), the users have to import the necessary CA certificates which requires manual validation of the certificates (for example using a published fingerprint).

A solution developed for fully automatic building of trust paths between PKIs are so called "cross certificates". A cross certificate is a certificate issued by a CA of one PKI, containing the public key and subject name of a different CA (usually from a different PKI). In this way the CA states that the key and subject name really belongs to the according certification authority. Because the cross certificate is issued by an already trusted CA, the process avoids the manual validation of CA certificates from other PKIs.

Issuing a cross certificate for another CA does not mean that the policies of both CAs are equivalent, it is just a further affirmation that the key in the cross certificate actually belongs to the named CA. This immediately raises the question how the certificates are validated by the application. If a certificate is always trusted if a trust path from it to an already installed root certificate exists, then a cross certificate (and all certificates issued by the named CA in it) would be trusted independent of the policy of the named CA. An application must be able to do a more detailed validation of certificates, if cross certs are to be used. The IETF PKIX-charter [RFC 3280] defines the usage of some special X.509 v3 extensions that specify equivalence of policies to automatise these decisions (eg. PolicyMappings, PolicyConstraints, PolicyQualifiers). The path validation algorithms are not required to support policy mappings according to [RFC 3280], but they must reject a certificate that contains an unsupported critical extension.

Another problem of the validation process is that cross certificates are manifold certificates: They have the same subject name than that of another certification authority. The usual trust
relation of all certificates is a set of trees (from a root certificate to the single certificates issued by subordinate CAs), one tree for each PKI. The integration of manifold certificates leads to multiple connections between the nodes of the graph, which may even introduce loops in the trust paths. The following figure shows the effects of different types of manifold certificates to the trust relation graph. Table 1 explains the different types of manifold certificates.

![PKI Connecting Public-Key-Infrastructures](image)

<table>
<thead>
<tr>
<th>Prolonging Certificates:</th>
<th>A new certificate is issued containing the same data, but a prolonged expiry date (and usually a new serial number).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exchange Certificates:</td>
<td>A new certificate is issued containing the same data, but with a modified set of extensions. This could be due to changes in standards or applications that require certain X.509 extensions.</td>
</tr>
<tr>
<td>Cross Certificates:</td>
<td>A public key is already contained in a certificate issued by a certification authority. A new certificate is issued by another certification authority containing the same key, subject and DN. The following types of cross certificates are possible:</td>
</tr>
<tr>
<td>Root signs Root</td>
<td>The linking of two complete PKIs.</td>
</tr>
<tr>
<td>Root signs SubCA</td>
<td>Linking only a subtree of one PKI to the other. This is usually done to link only a part for a single purpose to the own PKI, for example only the user certification authority, so only user certificates are included.</td>
</tr>
<tr>
<td>SubCA signs SubCA</td>
<td>Only parts of both PKIs are linked together, for example the user CAs of both.</td>
</tr>
</tbody>
</table>

*Table 1: Different Types of Cross Certificates*

While prolonging certificates and exchange certificates do only add double paths to already existing paths, cross certificates are the ones that create new edges with independent directions therefore enabling loops in the graph. A typical case for a loop in the trust graph is a root certificate change of a PKI. The old, expiring root certificate is used to sign the new key of the CA and the new one signs the old CA certificate (see [RFC 2510], [Ca++2002]).

Considering cross certificates and different CA policies the path validation becomes an even more complex process, which will not always be integrated into applications. New efforts are undertaken to establish server processes that can do the path validation for clients [RFC...
The concept of a Super Root CA can as well be implemented using cross certificates, so the CA certificates are not replaced by new ones, but an additional cross certificate for each CA is issued by the Super Root. From the perspective of an application, this procedure resembles the "Root signs SubCA" scenario, the only difference being that additionally the Super Root certificate has to be imported into the application.

Table 2 shows the scenarios that were chosen for the application tests.
<table>
<thead>
<tr>
<th>No</th>
<th>Graph</th>
<th>Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1.png" alt="Graph 1" /></td>
<td>Distribution of multiple root certificates together with key and certificate.</td>
</tr>
<tr>
<td>2</td>
<td><img src="image2.png" alt="Graph 2" /></td>
<td>Distribution of multiple root certificates in a single file.</td>
</tr>
<tr>
<td>3</td>
<td><img src="image3.png" alt="Graph 3" /></td>
<td>Using a Super CA issuing new CA certificates for each PKI to be linked. Old certificates are replaced.</td>
</tr>
<tr>
<td>4</td>
<td><img src="image4.png" alt="Graph 4" /></td>
<td>Cross Certificates: Root signs Root to integrate other PKI.</td>
</tr>
<tr>
<td>5</td>
<td><img src="image5.png" alt="Graph 5" /></td>
<td>Cross Certificates: Root signs SubCA to integrate part of other PKI.</td>
</tr>
<tr>
<td>6</td>
<td><img src="image6.png" alt="Graph 6" /></td>
<td>Cross Certificates: SubCA signs SubCA, only parts of both PKIs are linked.</td>
</tr>
</tbody>
</table>

Table 2: Linking Scenarios for the Tests

A seventh, special scenario deals with the root certificate update. Whenever a root certificate expires it is difficult to establish a trust relation to a new one. Root certificates are self-signed, so no signature validation leads to an already trusted certificate. Usually the certificate data (eg. the fingerprint) is either distributed on other communication paths (printed media, telephone lines) or included in new software releases (as in web browsers). A better method is

(c) 2004-2005 SURFnet and DFN-CERT Services
the root certificate update method described in [RFC 2510]. A new PKI and two additional
cross certificates are created. The cross certificates provide links between the old and the new
root certificates: one is the new root key, signed by the old root CA (new-with-old), the other
one is the old root key, signed by the new root CA (old-with-new). If the old root certificate is
already trusted, the cross-certificates establish a trust relation to the new root certificate. For
validation according to the shell model\(^1\) the certificate new-with-old should expire at the same
time as the old root certificate itself.

To decide which of these methods is to be applied to link different PKIs it is always necessary
to consider the applications that are part of the working environment. What use is an elegant
solution with cross-certificates if these are not supported by the clients? Usually detailed
information on the implemented PKI algorithms, the supported X.509 extensions etc is not
easy to get and may not always be sufficient or precise. This report should deliver some first
insights for a set of widespread email- and web-clients.

\(^1\) The shell modell means that the validity period of issued certificates should be fully embedded into the
validity period of the issuers certificate.
Establishing a Test Framework for PKI Linking Methods

For the testing of possible linking methods for different PKIs, a few prerequisite decisions have to be made. The kind of certificates to be used has to be specified as well as the way to generate them. The applications and operating systems to be tested have to be chosen and ways have to be found to perform the tests in a comparable and reproducible way. The following sections describe the design of the test framework.

General Concept

Two main kinds of applications make use of signing and encryption for authentication and privacy. One of them is electronic mail, the other is https for secure connections to servers. Both of them usually support the usage of PKI based on X.509 certificates. For each application class, different tests have to be conducted to test the support of the proposed PKI linking methods.

In the case of email clients, it was assumed that the handling of certificates is based on the same methods for signing, encrypting, decrypting and verifying. Based on this assumption only the validation of signed emails and encryption were tested (the other cases are more a matter of having the correct secret key than of chain validation of PKI certificate hierarchies). The characteristics of emails require only a very simple testbed: It is usually the data (the email itself) that is encrypted or signed, so it is sufficient to check an existing email. For encryption tests the email does not have to be sent, it is sufficient for PKI tests to actually do the encryption. Usually no further interaction with a networked environment is necessary. So if no special requirements of the software arise (like being not able to import a signed email), the testbed can be reduced to a single host running the email application to be tested.

Network connections between web clients and servers do have very different characteristics. In this case it is the connection to another host which has to be encrypted and authenticated. A complete client-server setup has to be established to do the testing. One host provides the application to be tested (client), another one provides the necessary web server to enable the establishment of connections. The testserver authenticates itself with a certificate from the foreign PKI. The primary test case is to establish a trustworthy connection to the web server. If this succeeds, the linking method between both PKIs works for the client. For the more complex certificate chains (containing cross certificates or additional CAs) more than one test my be applicable. For example in scenario 4, the chain from the server may contain only the old foreign PKI (server cert, sub CA cert & root CA cert), requiring the client to have the own root certificate and the cross certificate, or the serverchain may already contain the cross certificate (so the client does not need to have the cross certificate).
Creating the Certificates

First a profile for the certificates has to be specified. The profile should be simple and conservative to enable as many applications as possible to work with the generated certificates (otherwise the cross certificates, which are derived from the existing hierarchies, will not work for sure). A few X.509v3 extensions seem to complicate the import of certificates into applications, so some of the extensions specified in the PKIX standard [RFC 3280] were not used. A very basic certificate profile is the HEPKI PKI Lite profile for end users [HEPKI 2004b] and certification authorities [HEPKI 2004a] which serves as guideline for the used certificates. Appendix A shows the usage of the extensions specified by both profiles in the test certificates. For the conducted tests policy mapping extensions were not included in the certificates. This was chosen to always make sure that the test results are as simple and understandable as possible. More complex tests including policy mappings may prove necessary for further test projects, especially when the policy mapping extensions come into broad use.

To be able to publish the results and enable the reproduction of the tests, PKIs were setup for testing only. This way the keys and certificates used can be published without security concerns. For the creation and management of the X.509 certificates OpenSSL is used. Additionally some Perl scripts were created that produce standard test PKIs consisting of a root CA, a subordinate CA (subCA) which in turn issues some user and server certificates. For each linking method to be tested, a script exists that creates all required PKIs and exports the necessary keys and certificates into a few files (usually there are two of them, one for each of the email exchanging parties). This way the certificates can be easily imported into the client applications for testing. Documentation of the used scripts can be found in Appendix B.

To reduce the necessary test environment to a minimum, the scripts are also able to create signed S/MIME emails. These are written to a file in MBOX format, so most client applications should be able to read them as a local folder. This way it is not necessary to build a complete networked environment for creating, sending, receiving and validating signed emails. The MBOX file can simply be copied to the test host and the application must be configured to use the file as a folder. This way the signature validation test is reduced to the installation of the necessary certificates and checking the signature of the email.

In the case of web applications a server is needed to provide web connectivity. The primary test is to ask a client to establish a secure connection to a secure web server, so the server has to be authenticated. Usually the server provides the complete certificate chain and the client checks the certificates up to the root certificate, which must be already present in the browser. The proposed methods of PKI linking are either root certificate distribution (scenario 01 and 02) or more efficient methods to link new PKIs to existing PKIs. For the more complex scenarios the provided certificate chains may be changed on the client (eg importing not only a cross-certificate but also additional certificates) or on the server (the web server itself could provide a cross certificate to provide a trustpath to a PKI that is already integrated in the client). In these cases additional tests can be conducted. The focus of these tests is always on the capabilities of the client applications to handle the different linking methods.
Choosing the test applications

To make this report an instrumental document for decisions on how to solve the problem of linking real PKIs, the choice of tested applications was essential. The choice of operating systems was straightforward: The most common operating systems on today's users desktop are Microsoft Windows XP and Windows 2000. From the Unix operating systems, Linux was the environment of choice.

The focus on the users desktop also influenced the decision of the applications to be tested. In this first trial, the focus was set on client applications because these usually prove to be the stumbling stones in establishing a PKI. What use is a PKI when the users applications cannot handle certificates? The most important applications of X.509 certificates right now are email clients (using S/MIME emails) and web browsers (using SSL for server or client authentication). So the most common applications of these both domains were to be tested. Today's desktop usually runs a Microsoft operating system and the applications distributed with it. This makes Internet Explorer and Outlook Express the first choice of applications to be tested. For its broad use in commercial environments, Outlook XP was chosen as a third Microsoft application.

In the context of web browsers Mozilla was chosen as the most common complementary as well as its successor Firefox, last but not least because they are available in many different environments. The email application of Mozilla and the email counterpart of Firefox, Thunderbird both are very advanced email clients, supporting client authentication, S/MIME and PGP encryption.

The choice of further applications is more difficult because many programs can be taken into account. Kmail was chosen because of its portability (KDE runs on many different platforms) and because it uses the S/MIME support of GnuPG which seems to be one of the most advanced tools for encryption and signing. The web counterpart of KDE, konqueror, was the next included candidate.

In the context of web browsers, the next application on the list of widely used applications was Opera, again its ability to run in multiple environments made it a good choice for testing. After the seven test scenarios, the three operating systems and these 10 applications were chosen, it became clear that no further testing would be possible in the given time: 161 tests had to be conducted already.

Setting up the testbed

The testbed for all these tests was chosen to be as simple as possible. Only a single client host was used. It was essential to rule out any influences of the results by different applications (e.g. by certificates that remained in the certificate store or changed trust-settings). So Symantec's disk imaging software Ghost was used to produce images of the different operating systems each with only a single application installed (with two exceptions: Outlook Express and Internet Explorer were both installed in the same image as well as konqueror and kmail, but
even in these cases the tests were conducted with a single application from a fresh install). So on a second harddisk 17 images for each pair of application and operating system were installed. The following table lists all the used images (please note that image 3.1 does not exist):

<table>
<thead>
<tr>
<th>Image</th>
<th>Operating System</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Windows 2000 Pro SP4</td>
<td>Outlook Express 6.0 SP1, Internet Explorer 6.0 SP1</td>
</tr>
<tr>
<td>1.1</td>
<td></td>
<td>Outlook XP</td>
</tr>
<tr>
<td>1.2</td>
<td></td>
<td>Mozilla 1.7.2</td>
</tr>
<tr>
<td>1.3</td>
<td></td>
<td>Thunderbird 1.0</td>
</tr>
<tr>
<td>1.4</td>
<td></td>
<td>Firefox 1.0</td>
</tr>
<tr>
<td>1.5</td>
<td></td>
<td>Opera 7.54</td>
</tr>
<tr>
<td>2.0</td>
<td>Windows XP Pro SP2</td>
<td>Outlook Express 6.0 SP1, Internet Explorer 6.0 SP1</td>
</tr>
<tr>
<td>2.1</td>
<td></td>
<td>Outlook XP</td>
</tr>
<tr>
<td>2.2</td>
<td></td>
<td>Mozilla 1.7.2</td>
</tr>
<tr>
<td>2.3</td>
<td></td>
<td>Thunderbird 1.0</td>
</tr>
<tr>
<td>2.4</td>
<td></td>
<td>Firefox 1.0</td>
</tr>
<tr>
<td>2.5</td>
<td></td>
<td>Opera 7.54</td>
</tr>
<tr>
<td>3.0</td>
<td>SuSE Linux 9.2</td>
<td>kmail 1.7 &amp; konqueror (KDE 3.3)</td>
</tr>
<tr>
<td>3.2</td>
<td></td>
<td>Mozilla 1.7.2</td>
</tr>
<tr>
<td>3.3</td>
<td></td>
<td>Thunderbird 1.0</td>
</tr>
<tr>
<td>3.4</td>
<td></td>
<td>Firefox 1.0</td>
</tr>
<tr>
<td>3.5</td>
<td></td>
<td>Opera 7.54</td>
</tr>
</tbody>
</table>

For each test Ghost was used to install the necessary image on the first harddisk, from which the system was booted afterwards. This way the test environment was always a clean install of the operating system and the application to be tested.

With the installed image all the tests for a single application were done in sequence. The different keys used in each of the scenarios certificates ensured that no remains from a previous test could lead to a trusted certificate chain (eg if a certificate store was not completely cleared). At the beginning for each application it was tested if imported certificates really vanished when they were deleted in the application. No side effects between subsequent tests could be found.

For some email tests in the Linux environment, no test server was required. The simple format (mbox) of email folders supported by most Unix applications enabled the scripts to create folders with signed emails that could be copied into the inbox of the email application. Because of the undocumented and complex binary format that is used by the Microsoft email applications, a testserver was used which provided a POP3 service (for receiving email) and an SMTP service (for sending emails). On the server the mbox files generated by the scripts were copied to the mailbox of the configured testuser and were received by the tested
application using POP3.

The SSL tests were conducted with a HTTPS enabled Apache 2 web server on which the generated server certificates, keys and certificate chains for each scenario were installed. The Apache web server offers the possibility of directly influencing the certificate-chain that is transmitted by the server for server-authentication. This allows to do further testing on what order or presence of certificates the client application rely. This possibility is usually not given in other server software, so if a client requires a special certificate order, this could mean it will not work with some server software.
Tests

This chapter contains the descriptions of all tested applications as well as the results of the tests. For each PKI linking scenario the application tests are listed, email applications first followed by the web tests.

As observed after all the tests were completed, the results for those applications that were tested on different operating systems were the same, so for better readability and reduction of redundancy these tests were compiled into a single test description. This was the case for Mozilla, Thunderbird, Firefox and Opera.

The only exception is the import of PKCS #12 files using the Windows certificate manager. There exist different default settings for the PKCS #12 import for Windows 2000 and Windows XP. The differences are described in scenario 1 in the Outlook Express section.

Usually the first description of a test action (eg importing a PKCS #12 file using the Windows certificate manager) is described very detailed on the first occurrence and afterwards only a short summary and a reference to the detailed description is given.

The tests for the email application comprise three sections, the verification of a signed email (usually including the import of the necessary certificates and keys), the sending of an encrypted email and a short summary. The web tests only contain a server authentication (again including the import of the necessary certificates and keys) and the summary section.

All the results are summarized in the results chapter which also includes an overview table.
PKI-Linking-Report: Connecting Public-Key-Infrastructures

Test Scenario 1: PKCS12 Root Certificate Distribution

If a certification authority does not only certify the public keys of users but also generates the secret keys, a PKCS #12 file usually serves as medium for secure key distribution. The file contains the secret key of the user as well as the certified public key. PKCS12 is also able to hold additional certificates so the full chain of issuing CA certificates can be included. This mechanism enables the integration of different PKIs into the client applications, the PKCS #12 file can hold some additional root certificates, so that these are present in the users environment immediately after the import of the users certificate. Additional manual configuration may be necessary to make these root certificates work, for example Mozilla requires some trust settings for each CA regarding the purpose of the issued certificates (web server authentication, mail client authentication, object signing).

So for the test a PKCS #12 file is generated including not only the key and certificate of a user, but also the complete related certificate chain (subCA and root certificate) as well as an additional root certificate of the PKI to be linked. After the import of the file, both PKIs should be trusted by the application.

Email Test Preparations

For the tests two PKIs are created (PKI A and PKI B), each consisting of a root CA, a subCA and a user certificate. The script build_scenario_01.pl generates two PKCS #12 files and a signed email:

- scenario-01.PKI-B.p12: Contains a full set of data for PKI B: user-key, user-certificate, subCA certificate and root CA certificate. Importing this file enables the user B to generate signed emails (that can be tested by user A)

- scenario-01.PKI-A.p12: Contains a full set of data for PKI A (user-key, user-certificate, subCA certificate and rootCA certificate) and additionally the rootCA certificate of PKI B. Importing this file should enable the user A to verify the signed emails from user B.

- scenario-01.signedmail: Contains an S/MIME email signed by user B. In the PKCS #7 attachment is the signature of the mail and the complete certificate chain of PKI B. The file is already in MBOX format for easy testing in a local environment.

Two different tests are conducted. The first one is to import the file scenario-01.PKI-A.p12 into the client and then point the application to use the file scenario-01.signedmail as a mail folder (if this is not possible, the email is fetched using the POP3-protocol). The application can then process and display the email signed by user B. The signature of the testmail should be validated by the application and lead to the display of a trusted signed email.

The second test checks the ability of the client to use the certificates for encryption. Initially the key of the recipient has to be provided. Usually this already happens on receipt of the first signed email from the recipient. For the tests an encrypted mail to the sender of the signed email (from the previous test) is generated. In a larger environment additional mechanisms for key distribution (like LDAP) would be necessary.
**Web Test Preparations**

For the WWW tests, the script `build_scenario_01.pl` creates some additional files:

- **scenario-01.PKI-B-servercert.pem**: The certificate for a web server of PKI B. This must be configured using the Apache option `SSLCertificateFile`.

- **scenario-01.PKI-B-serverkey.pem**: The server's secret key which belongs to the installed server certificate. This file must be configured using the option `SSLCertificateKeyFile`.

- **scenario-01.PKI-B-serverchain.pem**: A certificate chain containing the certificates of the server CA and the root CA of PKI B. This file must be configured using the option `SSLCertificateChainFile`.

To test the server authentication, the server key, certificate and the chainfiles of PKI A should be installed on the webserver. For the server certificate to work, it must have a correct DNS name as CN (variable `dn_server_b.CN` in the script's configuration file). The name should resolve to the IP address of the test server (on a Unix system, an entry in `/etc/hosts` should be sufficient, Windows XP and Windows 2000 use the file `C:\windows\system32\drivers\etc\hosts`).

The PKCS12 file `scenario-01.PKI-A.p12` is installed in the web client. After the import of the certificates, the root CA certificates of PKI A and B should exist in the list of trusted CAs (if the import fails, the rootCA certificates can be installed manually from the folder `single_certs`). Additional configuration of trust settings may be necessary to allow the root CA B (and eventually the subCA B) to certify web servers. Requesting a document from the testserver (by typing the according URL with protocol https://) should now lead to a trusted connection.

**Application Tests**

The following sections present the test results of each tested application.

**Email Application: Outlook Express 6.0 SP1, OS: Windows 2000, Windows XP**

**Signature validation**: Outlook Express uses the Windows Certificate Manager to handle X.509 certificates. To import the personal key and certificates contained in a PKCS #12 file the "InternetOptions" in the system settings have to be opened. From the tab "content" the certificate manager can be started. The import of a PKCS #12 file is straightforward, from the "own certificates" tab, choose import, the certificate manager asks for the file name, the passphrase of the PKCS #12 file and the certificate store to be used. Here is the only important difference to be found between Windows 2000 and Windows XP: The default for Windows 2000 is to choose automatically, where to store the certificates. This way, the personal certificates are imported into the store of the own certificates, while the contained CA certificates are stored into the intermediate CA store and the trusted authorities store. The default for Windows XP is to store the imported certificates in the store belonging to the
actual certificate manager tab. In this case this makes Windows XP import the user certificates into the chosen "own certificates" store, while the CA certificates are not imported at all (without warnings). So for successfully importing a PKCS #12 file that contains CA certificates, the option "choose certificate store automatically" has to be selected.

In the final step of doing the import, the certificate manager presents each root certificate contained in the file and asks the user if he wants to trust this PKI:

After installing both contained root certificates as trusted, a signed mail (received using POP3) is displayed as trusted:

So the import of multiple root certificates contained in a PKCS #12 file worked.

**Encryption:** On validating the received signed mail, the certificate of user B is imported automatically into the certificate manager. This way it is possible to write an encrypted mail to this user without additional means (eg. fetching the key using LDAP). Outlook Express automatically chooses the own certificate of the sender (probably choosing the certificates for which a secret key exists), so the email is encrypted for user B and the local user A.

**Summary:** Outlook Express imports multiple certificates distributed in a PKCS #12 file. Each root certificate is presented to the user who has to decide if he trusts it. After receiving of a signed mail, encryption to the senders key is possible without further user interaction.
Email Application: Outlook XP Pro SP2, OS: Windows 2000, Windows XP

Signature validation: On the first glance Outlook XP looks very similar to Outlook Express. It uses the certificate manager of Windows for the X.509 functionality, so the handling of certificates should be the same. But obviously Outlook XP has additional X.509 functionality which results in a different handling for encrypted emails (see below). The necessary steps for signature validation and import of own keys and certificates are the same as for Outlook Express, so the import of the PKCS #12 is exactly the same (see scenario 1, Outlook Express). Again each contained root certificate has to be acknowledged by the user. After the installation of the PKCS #12 file and the acknowledgment of the included root certificates of PKI A and PKI B, a signed email from user B of PKI B is validated successfully:

So the import of multiple root certificates contained in a PKCS #12 file worked.

Encryption: Choosing to encrypt an email is not very obvious, the selectors for encryption and signing are hidden in message options -> security settings. It is possible to install according buttons in the tool bar of the edit window though. The first attempt to send an encrypted email failed: Outlook XP displays a generic error message which states that either the certificate does not exist or is invalid or the encryption algorithms are incompatible or not supported. A look into the certificate manager confirms that the certificate of user B was not imported on receiving his signed email. Outlook XP requires the user to import all certificates manually into the contact database. In the contact manager, there is a tab for a users certificate which can be installed from a file. It is possible to import the certificate directly from a signed email of this user. This is done by selecting the "from" field of a displayed email, calling the context menu (right mouse click) and choosing the entry "add to contacts".

Page 22 of 92 (c) 2004-2005 SURFnet and DFN-CERT Services
Not only the name and the email address of the sender are imported into the contacts but also his certificate. When the new contact is saved to disk it is possible to send encrypted emails to the user. One would expect the sent email to be encrypted to the recipient (user B) and to the sender (user A), which is indeed the case. Surprisingly Outlook XP displays the email in the sent folder only as "encrypted to user A".

A cross check using openssl to decrypt the email with the key of user B revealed that the email is correctly encrypted to both users.

**Summary:** Outlook XP imports multiple certificates distributed in a PKCS #12 file. Each root certificate is presented to the user who has to decide if he trusts it. After receiving of a signed email, the user is required to add the sender to his contacts database, otherwise the senders certificate is not imported. Only after this procedure it is possible to send encrypted emails.

**Email Application:** Mozilla 1.7.2, **OS:** Windows 2000, Windows XP, Linux

**Signature validation:** The import of a PKCS #12 file is done using the certificate manager of Mozilla, which leads a very hidden life in the preferences (*Edit -> Preferences -> Privacy & Security -> Certificates -> Manage Certificates*). Mozilla asks for the file name to import and the import password and imports the PKCS #12 file. All contained root certificates are imported into the list of known authorities, but initially no trust is given to them.
A signed email (received via POP3) is still displayed as having an invalid signature:

In the certificate manager, the trust of the imported CA certificates can be changed manually (by selecting the button "edit"). Mozilla offers to trust the CA for three different purposes ("identify websites", "identify mail users" and "identify software makers"). After enabling root CA B to identify mail users, the signature of the signed mail can be validated. So the import of multiple root certificates contained in a PKCS #12 file works for Mozilla, but it requires to edit the trustlevel of each imported root certificate.

It seems that Mozilla only imports certificates from emails if the according root-certificate is already trusted and the email is still new (= unread). So it is necessary to edit the trustlevel of root CA B before a signed email of user B is processed. Otherwise the intermediate certificate of subCA B is not imported and the certificate chain cannot be validated.

**Encryption**: While writing an email, encryption and signing can be chosen from the S/MIME menu in the toolbar. If no own certificate is configured, Mozilla refuses to encrypt the email, because it is not possible to encrypt it for the sender. The locally imported key from the PKCS #12 file can be selected in Edit -> Mail & Newsgroup Account Settings -> UserAccount -> Security. Afterwards Mozilla encrypts emails flawlessly.

**Summary**: Mozilla is able to import multiple CA certificates from a PKCS #12 file. After the import, each of them has to be configured manually to be trusted for mail (and web, if required). Mozilla only imports certificates contained in signed emails when the related root certificate is already installed and trusted, so emails cannot be used to provide new root
Email Application: Thunderbird 1.0, OS: Windows 2000, Windows XP, Linux

Thunderbird is of course still very similar to Mozilla. Indeed there is no difference in the behaviour of Thunderbird for this test. The import of multiple root certificates within a PKCS #12 file works, for each root certificate the trust settings have to be edited after the import. For further details see the description of Mozilla above.

Email Application: Kmail 1.7.1 (KDE 3.3.0), OS: Linux

Signature validation: In a first step kmail was configured to read email from a local mailbox. The generated signed S/MIME mail was copied to the configured local mailbox file (/var/mail/test) and "received" by kmail. The mail is displayed as not verified as shown below.

On selecting the "Details" link, kmail opens kleopatra, the certificate manager of GnuPG. It displays the included certificate chain correctly and shows the certificate as "not validated" (indicated by yellow textcolor in the image below).
The reason is obvious: The root certificate is not configured to be a trusted one. Surprisingly after importing the root certificate of PKI B using the kde privacy and security manager (in the kcontrol management tool), the status of the certificate does not change (even after restarting the applications). There exists no option to configure the trustlevel of certificates in the kleopatra application. Importing the complete certificate chain into kleopatra from a PKCS #12 file does not help either (which is logical, all the necessary certificates are already there), but gives a first hint on where to search for the solution: Importing of secret keys from a PKCS #12 file requires the gpg-agent.

The certificate manager kleopatra does not search the usual certificate store of kde, it only trusts GnuPG to provide valid certificate information. Right now there is no documentation on how to install trusted root certificates into GnuPG (this fact is stated in the TODO file of the gpg2 package). There are two ways to do the job: Either the certificate fingerprint (SHA1) is entered manually into the according gnupg configuration file (~/.gnupg/trustlist.txt), or the gpg-agent (which handles passphrases and keys) has to be started with the option "--allow-mark-trusted". The second method enables the gpg-agent to query the user if the required root certificate is trustworthy (as shown in the following dialog).

In a second dialog the gpg-agent queries for the trusted usage of the new root certificate.
After marking the root certificate of PKI B as trusted, **kmail** shows the email as validly signed.

The validation process of **kleopatra** shows the certificates according to their status: The root certificate is marked as "trusted root certificate" (though the configuration of **kleopatra** seems to be German only: "Vertrauenswürdiges Wurzelzertifikat"), in our configuration indicated by the color blue.
Encryption: To encrypt an email to a recipient of PKI B, a user from PKI A needs to get the key of the recipient first. Keys can be fetched by receiving a signed mail from the key owner, by querying an LDAP server or by manual distribution of certificate files. For encryption \textbf{kmail} requires an own certificate and secret key to be configured (even if the option "encrypt to self" is not set), so \textbf{kleopatra} was used to import the certificate chain and secret key of a local user of PKI A (using the PKCS #12 file). If the option is set, the local CAs root certificate (PKI A) must be marked as trusted, too. After the key import, own keys for signing and encryption can be selected in the \textbf{kmail} configuration menu. On replying to the signed email from the signature validation test and selecting encryption, \textbf{kmail} asks the user to approve the selected key for the recipient (see image below) and encrypts the mail successfully.

If the certificate chain of the recipient was not received in a previous signed email, it can be imported into \textbf{kleopatra} manually. If the root certificate is not trusted, gpg-agent asks the user to approve its fingerprint before the email is encrypted. For encryption to work, the full recipients certificate chain has to be supplied (root certificate, intermediate CA certificates, end user certificates). This means in order to enable a user of PKI A to send encrypted emails to users of PKI B, a PKCS #12 file does not provide a scalable solution, because it is not appropriate to store thousands of certificates). Additional services like LDAP (which seems to be supported by \textbf{kleopatra}) could solve this problem but were not tested.

\textbf{Summary:} The mailer \textbf{kmail} uses the certificate chains included in the transmitted S/MIME emails. It is not necessary to distribute the certificates by other means (as scenario 1 intended). The trust level of root certificates still has to be edited by the user, in larger setups a preconfigured \textbf{trustlist.txt} can be provided. In order to send encrypted emails either a signed email from the designated recipient can provide his complete certificate chain (including his end user certificate) or LDAP solutions need to be applied.

\textbf{Web Application: Internet Explorer 6.0 SP1, OS: Windows 2000, Windows XP}  
\textbf{Server Authentication:} A first try to establish a secure connection to the testserver produces the usual warnings about an untrusted certificate. The reasons for the warnings are explained
quite detailed, in this case the browser states that the certificate was issued by an untrusted organisation. The certificate chain can be displayed and shows that the server certificate is invalid:

On selecting the servers certificate, the browser displays the state of the certificate (claiming that the certificate cannot be validated up to a trusted authority). Surprisingly on selecting the root-certificate, the browser states that the certificate is valid. Obviously this means that the root-certificate is not expired or revoked, but it does not mean that the root-certificate is trusted.

Importing multiple root certificates in a PKCS #12 file works using the Windows certificate manager (see the description of Outlook Express above). The internet options can be reached not only by selecting the system settings in the start menu, there is also an entry in the "Extras" menu of Internet Explorer. The user is asked to acknowledge the trust settings for each root-certificate contained in the PKCS #12 file.

A connection to the testserver can then be established without warnings. The connection is displayed as being secure (the icon of a little lock in the lower right of the Explorer window: 

Summary: Internet Explorer supports the import of multiple root certificates very well. This way trust paths to external PKIs can be established when internal user certificates are distributed. The user has still to acknowledge each contained root certificate.

Web Application: Mozilla 1.7.2, OS: Windows 2000, Windows XP, Linux

Server Authentication: Import of PKCS #12 files is of course the same as with Mozilla Mail (see above), all contained root-certificates are imported, but each certificates trust settings have to be edited afterwards. After the root-certificate of PKI B is set to "This certificate can identify websites", a secure connection to the testserver can be established without warnings. Mozilla displays a small icon with a lock in the lower right of the window. By clicking on the icon, Mozilla opens the following information window:
Summary: Mozilla is able to import multiple CA certificates from a PKCS #12 file. After the import, each of them has to be configured manually to be trusted for identification of websites (and email users if required).

Web Application: Firefox 1.0, OS: Windows 2000, Windows XP, Linux
Firefox is still very similar to Mozilla. Indeed there is no difference in the behaviour of Firefox for this test. The import of multiple root certificates within a PKCS #12 file works, for each root certificate the trust settings have to be edited after the import. For further details see the description of the Mozilla web browser above.

Web Application: Opera 7.54, OS: Windows 2000, Windows XP, Linux
Server Authentication: A first try to establish a secure connection to the testserver produces the usual warnings about an untrusted certificate. The full certificate-chain provided by the webserver is displayed. For each of the certificates, detailed information of the certificate state and the reasons of failure are displayed:
Certificates can be imported using the Opera certificate manager (which is buried deep in the menus in Tools -> Preferences -> Security -> Manage Certificates). On importing a PKCS #12 file, Opera displays all contained certificates and asks the user, if these should be imported. There is no way of selecting only some of the presented certificates.

During the import, Opera seems to try to check if the imported certificates belong to an existing authority. Obviously Opera fails to build the path to the root certificate of PKI A (which is included in the PKCS #12 file) so it displays a warning (see figure below). This seems a little bit odd and the warning can be (or better: must be) ignored. Order of the certificates in the PKCS #12 file is relevant to Opera.
After the import, the list of trusted authorities does not include the imported root-certificates, only after exiting and reopening of the certificate-manager the updated list is displayed. For each certificate Opera has two trust settings. The first is to allow a certificate to be used for authentication and the second is to display or not to display warnings when the certificate is used. The default settings for the imported CA certificates from the PKCS #12 file are a little bit odd: Both certificates of the local PKI (rootCA A and subCA A) are allowed for authentication, but warnings are switched on. The root-certificate of the foreign PKI (rootCA B) is allowed with no warnings. Usually the trust settings for the own PKI will have to be edited manually after the import. For our test connection to the server of PKI B, no manual changes are necessary after the import of the PKCS #12 file, establishing a trusted connection works without warnings. Opera does not display too much information about the trusted connection, the only status information is the bubble help of the lock symbol:

**Summary:** Opera supports the import of multiple root-certificates from a PKCS #12 file. After the import it may be necessary to edit the trust settings for each imported certificate to get rid of the warnings when a secure connection is established.

**Web Application: Konqueror 3.3 (KDE 3.3.0), OS: Linux**

**Server Authentication:** A first connection establishment to the test server shows the usual alert: "The server certificate failed the authenticity test". The details of the complete certificate chain provided by the server can be displayed. Here the reason for the failure is listed: "Certificate is self-signed and thus may not be trustworthy."
In the cryptography configuration, the PKCS #12 file can be imported. The import works without complications, but it seems that only the enduser certificate and key is imported, not the additional certificates. So the root certificate of PKI B is not imported as well as the certificate chain of the users own PKI A.

In the configuration tab "SSL Signers", CA certificates can be imported manually. DER, PEM and Netscape encoded certificates are supported. After the import of the root certificate of PKI B, the server authentication succeeds:
PKI-Linking-Report: Connecting Public-Key-Infrastructures

So the trial to provide additional root certificates while delivering the clients own certificate fails due to the restricted import functionality. A manual import of the trust anchor is still possible. In larger setups, the root certificates could be distributed in the global KDE configuration file `/opt/kde3/share/config/ksslcalist` or in the users file `~/.kde/share/config/ksslcalist`. The file contains the PEM encoded certificates and the trust values for email signing, SSL servers and object signing in ASCII format.

**Summary:** Konqueror does not import all provided certificates from a PKCS #12 file, so the distribution of additional certificates together with the endusers secret key and certificate is not possible. By manual import of the root certificates or installation in the according configuration files the trust to other PKIs can be established.
Test Scenario 2: PKCS7 Root Certificate Distribution

To enable the local applications to validate signatures from different PKIs, all the root certificates of the PKIs have to be installed in each application. To provide a default set of trusted CA certificates, the root certificates can be bundled into one file that is imported by the application. Usually PKCS #7 files serve as a medium to distribute certificates. So a file is created containing the root certificates of different PKIs. After the import of the certificate bundle, signature validation should work for all the contained PKIs. Additional manual configuration may be necessary to make these root certificates work (eg. editing the trust settings for each CA).

Email Test Preparations

For the tests two PKIs are created (PKI A and PKI B), each consisting of a root CA, a subCA and a user-certificate. The script build_scenario_02.pl generates a PKCS12 files, a PKCS7 file and a signed email:

- scenario-02.PKI-B.p12: Contains a full set of data for PKI B (user-key, user-certificate, subca-certificate and rootca-certificate). Importing this file enables the user B to generate signed emails (that can be tested by user A).
- scenario-02.PKI-linking-A.p7: Contains the root certificates of PKI A and PKI B. Importing this file should enable the user A to verify the signed emails from user B.
- scenario-02.signedmail: Contains an S/MIME email signed by user B. In the PKCS7 attachment is the signature of the email and the complete certificate chain of PKI B. The file is already in MBOX format for easy testing in a local environment.

Additionally all CA certificates are exported to single DER files. This enables detailed testing against the import of certificate bundles (eg to make sure, that each single certificate does work with the application, even if the import of the certificate bundle fails).

The test can be conducted in two different ways. The first one is to import the file scenario-02.PKI-linking-A.p7 into the client and then point the application to use the file scenario-02.signedmail as an email folder. The signature of the test email created by user B from PKI B should be validated by the application and lead to the display of a trusted signed email.

The second way to conduct this test (and maybe some further examinations) is to use a full email setting including a sending client, an email environment and a receiving client. On the sending side, the file scenario-02.PKI-B.p12 should be imported into the client application. Signed emails from PKI B can now be produced using the application. On the receiving side, scenario-02.PKI-linking-A.p7 must be installed. Emails from the sender can now be received and validated.

If the import of the PKCS7-file scenario-02.PKI-linking-A.p7 is not possible because of problems with file name suffixes, an identical file is created using the name scenario-02.PKI-linking-A.der to ease the import of the file. If it still does not work, each single CA certificate should be imported into the application to make sure that the problems are due

(c) 2004-2005 SURFnet and DFN-CERT Services
to the import of the complete certificate bundle and not caused by the single certificates (eg due to special X.509 extensions).

**Web Test Preparations**

For the web tests, the script `build_scenario_02.pl` creates the usual files (server certificate, key and chain for PKI B, see Scenario 1), which must be installed on the test web server. After import of the file `scenario-02.PKI-linking-A.p7` into the client, a secure connection should be established without the usual browser dialogs concerning untrusted certificates. Additional configuration of trust settings may be necessary to allow the root CA B (and eventually the subCA B) to certify webservers.

**Application Tests**

The following sections present the test results of each tested application.

**Email Application: Outlook Express 6.0 SP1, OS: Windows 2000, Windows XP**

**Signature validation:** The import of a PKCS #7 file is done with the Windows certificate manager, the import is similar to the PKCS #12 import of scenario 01. In this case only trusted authorities are imported from the PKCS #7 file, so the import is started from the "trusted authorities" tab of the certificate manager. Windows expects the PKCS #7 file to have the suffix ".p7b", otherwise it is not listed in the file browser. The import does not require a password because the file contains no secret key. Windows 2000 defaults to choose the certificate store automatically, in Windows XP the default is to use the chosen certificate store (so it's the "trusted authorities" store which is correct in this case). Each contained root certificate is presented to the user for acknowledgment. After installation of both contained root certificates a signed mail is displayed as trusted.

**Encryption:** If no own private key exists, Outlook Express informs the local user (user A) that an email encrypted to user B cannot be read again, but encryption is possible without an own key. If a PKCS12 file is imported providing a key and certificate for user A, encryption works without queries as in scenario 1.

**Summary:** Outlook Express imports multiple certificates distributed in a PKCS7 file. Each root certificate is presented to the user who has to decide if he trusts it. After receiving of a signed mail, encryption to the senders key is possible without further user interaction.

**Email Application: Outlook XP Pro SP2, OS: Windows 2000, Windows XP**

**Signature validation:** The import of certificates and keys for signature validation is done using the Windows certificate manager, so it is exactly the same as the PKCS #7 import with Outlook Express (see scenario 2, Outlook Express). The import works as expected, both contained root certificates have to be acknowledged by the user. Now the signature from user
B can be validated.

**Encryption**: Again encryption requires to import the recipients certificate as described before (see scenario 1, Outlook Express). Again, after encrypting the email, Outlook states that the email is encrypted for the local user (sender) only.

**Summary**: The import of multiple root certificate using a PKCS #7 file works as expected, each contained root certificate is presented to the user for acknowledgment. This enables to import multiple trusted root certificates in a convenient way.

**Email Application: Mozilla 1.7.2, OS: Windows 2000, Windows XP, Linux**

**Signature validation**: The import of a PKCS #7 file is done using the certificate manager of Mozilla (found in *Edit -> Preferences -> Privacy & Security -> Certificates -> Manage Certificates*). Mozilla asks for the file name to import expecting file suffixes ".cer", ".crt", ".cert", ".der" or ".pem". If the PKCS #7 file is selected for import, Mozilla asks the user to edit or acknowledge the trust settings of the first contained root-certificate and then states, that the import was successfull. A look into the list of imported authorities shows that only the fist certificate from the PKCS #7 file was imported.

This behaviour is a little bit different from that reported in earlier versions (see the discussion under [https://bugzilla.mozilla.org/show_bug.cgi?id=236461](https://bugzilla.mozilla.org/show_bug.cgi?id=236461) and [http://www.terena.nl/mail/archives/tf-aace/msg00654.html](http://www.terena.nl/mail/archives/tf-aace/msg00654.html)), but reflects more or less the opinion of the developers, that a PKCS #7 file should not contain "a collection of potentially unrelated certs". So PKCS #7 files are no solution for distributing different root certificates to Mozilla users.

Of course all root certificates can be imported stepwise using single DER- or PEM-encoded certificates and setting the trustlevel accordingly. Afterwards signature validation works as already described in scenario 1, Mozilla Mail.

**Encryption**: After the import of the root-certificates, Mozilla Mail still needs the local users certificate to be installed, otherwise it refuses to generate encrypted mails. After importing an own key and certificate (using PKCS #12) and configuring the certificate as "own certificate" (in *Edit -> Mail and Newsgroups Account Settings -> UserAccount -> Security*) encryption works as expected.

**Summary**: Mozilla is not able to import multiple root certificates from a PKCS #7 file. So scenario 2 is not suitable to establish trust in multiple PKIs for Mozilla Mail environments.

**Email Application: Thunderbird 1.0, OS: Windows 2000, Windows XP, Linux**

In scenario 2 Thunderbird behaves exactly the same as Mozilla 1.7.2 in all three operating systems: PKCS #7 files needs the expected filename suffix (eg. ".der") to be listed in the import browser, only the first root certificate is imported. For further details see the description of Mozilla Mail above.
Email Application: Kmail 1.7.1 (KDE 3.3.0), OS: Linux

Signature validation: Kmail was configured as shown in scenario 01, the signed email is displayed as not verified. The import of the PKCS #7 file containing the root certificates is supported by the keymanager kleopatra and gpgsm (kleopatra expects the file suffix ".p7m"), but the imported certificates are not verified and not marked as trusted (indicated by the purple color of the entries in the figure below).

![Kmail keymanager](image)

The root certificates may be validated by the gpg-agent (eg when a signed email is selected in kmail) or by entering the fingerprint in ~/.gnupg/trustlist.txt. Afterwards kmail displays the mail with a green header and a status "Good Signature".

Encryption: Encryption works as described in scenario 1.

Summary: Kmail uses the certificate chains included in the received S/MIME emails. Importing of complete sets of root certificates can be achieved using gpgsm or kleopatra. The trust level of root certificates still has to be edited by the user, in larger setups a preconfigured trustlist.txt can be provided. In order to send encrypted emails either a signed email from the designated recipient can provide his complete certificate chain (including his end user certificate) or LDAP solutions need to be applied.

Web Application: Internet Explorer 6.0 SP1, OS: Windows 2000, Windows XP

Server Authentication: Using the Windows certificate manager PKCS #7 files can be used to import new root-certificates into the list of trusted authorities. Again windows expects these files to have a filename suffix of ".p7b":

![Windows certificate manager](image)

The user is asked to verify each contained root-certificate (using the displayed fingerprint). After the import of the contained root-certificates of PKI A and PKI B, secure connections can be established to the test server without warnings.
Summary: Internet Explorer supports the import of multiple root-certificates from a PKCS #7 file. This way trust paths to external PKIs can be established when internal user certificates are distributed. The user has still to acknowledge each contained root certificate.

Web Application: Mozilla 1.7.2, OS: Windows 2000, Windows XP, Linux

Server Authentication: Import of PKCS #7 files is the same as with Mozilla Mail (see above), only the first root certificate is imported into the browser and no error message is displayed. All root-certificates can be imported as single ".DER"-files thereafter server authentication works as expected.

Summary: Mozilla is not able to import multiple CA certificates from a PKCS #7 file. Hence scenario 2 is not suitable to establish trust in multiple PKIs for Mozilla environments.

Web Application: Firefox 1.0, OS: Windows 2000, Windows XP, Linux

In scenario 2 Firefox behaves exactly the same as Mozilla 1.7.2 in all three operating systems: PKCS #7 files needs the expected filename suffix (eg. ".der") to be listed in the import browser, only the first root certificate is imported. For further details see the description of the Mozilla web browser above.

Web Application: Opera 7.54, OS: Windows 2000, Windows XP, Linux

Server Authentication: Opera supports the import of root-certificates from a PKCS #7 file. Certificates can be installed using the Opera certificate manager which expects a filename suffix of ".p7". On importing the chosen file, Opera displays all contained certificates and asks the user if these should be imported. There is no way of selecting only some of the presented certificates.

(c) 2004-2005 SURFnet and DFN-CERT Services
Warnings are switched on by default for the imported certificates. This has to be changed manually for each certificate (the trust settings are opened by selecting the certificate from the list of trusted authorities and clicking on the "View" button). After switching off the warnings a trusted connection to the testserver can be established without warnings.

**Summary**: Opera supports the import of multiple root-certificates from a PKCS #7 file. After the import it is necessary to edit the trust settings for each imported certificate to get rid of the warnings when a secure connection is established.

**WWW-Application: Konqueror 3.3 (KDE 3.3.0), OS: Linux**

**Server Authentication**: The first connection attempt leads to the same results as the test in scenario 1. The import of the prepared PKCS #7 file does not work, obviously PKCS #7 is not supported by the KDE environment. This behaviour is independent from the suffix of the filename, even if the file is named with the suffix "*.der", the import fails.

![Certificate File Not Loadable](image)

As already tested in scenario 1, the manual import of single certificates works as expected, in larger setups CA certificates can be provided in the according KDE files (see Konqueror tests scenario 1).

**Summary**: Konqueror does not support certificate import from a PKCS #7 file, so the distribution of a collection of root CA certificates is not feasible this way. By manual import of the root certificates or installation in the according configuration files the trust to other PKIs can be established.
**Test Scenario 3: SuperCA PKI Federation**

To build a federation of different PKIs a SuperCA can be created, that will present the new, single root node of all linked PKIs. The SuperCA issues new certificates for each former root CA, so the root CAs become SubCAs of the SuperCA. After the creation of this new PKI structure, the new certificates of the SuperCA and its SubCAs have to be distributed to the client applications.

**Email Test Preparations**

For the tests two PKIs are created (PKI A and PKI B), each consisting of a root CA, a subCA and a user certificate. From PKI B a complete set of a users certificates (containing key, certificate and certificate chain) is exported as PKCS #12 file. A signed email is created using the actual certificate chain of PKI B. After this, the new SuperCA is created and both PKIs become SubCAs of it. The new certificates of the SuperCA and both SubCAs are exported to a PKCS #7 file. Again a signed email is created, now containing the complete certificate chain of PKI B (including the certificate of the SuperCA). A second PKCS #12 file is generated holding the users key and certificate as well as the new complete chain of PKI B (including the SuperCA certificate and the new SubCA certificate of the former root CA of PKI B). The script `build_scenario_03.pl` generates two PKCS #12 files, a PKCS #7 file and two signed emails:

- `scenario-03.PKI-B-before.p12`: Contains a full set of data for PKI B (user-key, user-certificate, subCA certificate and rootCA certificate). Importing this file enables the user B to generate signed emails (that can be tested by user A).

- `scenario-03.signed-before`: Contains an S/MIME email signed by user B. In the PKCS7 attachment is the signature of the mail and the complete certificate chain of PKI B (subCA certificate and rootCA certificate). The file is already in MBOX format for easy testing in a local environment.

- `scenario-03.PKI-B-after.p12`: Contains a full set of data for PKI B (user-key, user-certificate, subCA certificate, new rootCA certificate and SuperCA certificate). Importing this file enables the user B to generate signed emails (that can be tested by user A).

- `scenario-03.link-certs.p7`: Contains the new SubCA certificates of PKI A and PKI B as well as the certificate of the SuperCA. Importing this file should enable the user A to verify the signed emails from user B. For applications requiring the suffix ".der" for the certificate import, an identical file `scenario-03.link-certs.der` is created.

- `scenario-03.signed-after`: Contains an S/MIME email signed by user B. In the PKCS7 attachment is the signature of the mail and the complete certificate chain of the new PKI B (including the new SubCA certificate of the former rootCA B and the SuperCA certificate). The file is again in MBOX format.

Additionally all CA certificates are exported to single DER files to circumvent difficulties with the import of certificate bundles.

(c) 2004-2005 SURFnet and DFN-CERT Services
The test can be conducted in two different ways. The first one is to import the file `scenario-03.link-certs.p7` (or the according single certificates) into the client and then point the application to use the file `scenario-03.signed-before` as an email folder. The signature of the test email created by user B from PKI B should be validated by the application and lead to the display of a trusted email signature. If the validation fails, a further test can be conducted using the second email file `scenario-03.signed-after`. This email contains the complete new certificate chain up to the SuperCA. In both cases it may be necessary to edit the trust settings for the SuperCA.

The second way to conduct this test (and maybe some further examinations) is to use a full email setting including a sending client, an email environment and a receiving client. On the sending side, the file `scenario-03.PKI-B-before.p12` should be imported into the client application. Signed emails from PKI B can now be produced using the application. On the receiving side, `scenario-03.link-certs.p7` must be installed. Emails from the sender can now be received and validated. If this fails, the second PKCS12 file can be installed in the sender application to see if signature validation works, when the new full certificate chain is included in the mail.

If the import of the PKCS7-file `scenario-03.link-certs.p7` is not possible because of the necessary file name suffixes, an identical file is created using the name `scenario-03.link-certs.der` to ease the import of the file. If it still does not work, each single CA certificate should be imported into the application to make sure that the problems are due to the import of the complete certificate bundle and not caused by the single certificates (e.g. due to special X.509 extensions).

**Web Test Preparations**

For the web tests, the script `build_scenario_03.pl` creates the usual files (server certificate, key and chain for PKI B, see Scenario 1), which must be installed on the test web server. After import of the file `scenario-03.link-certs.p7` (or, if this fails, the manual import of the SuperCA certificate and the subordinate CA certificates issued to both rootCAs) an SSL connection should be established without the usual browser dialogs concerning untrusted certificates. Additional configuration of trust settings may be necessary to allow the root CA B (and eventually the subCA B) to certify web servers.

In the first test, all the necessary changes to integrate PKI B are made on the client, the certificate chain provided by the server stays the same (server B, subCA B & rootCA B). An additional test can be performed by only importing the SuperCA certificate into the client and configuring the server to provide the chain server B, subCA B, rootCA B (issued by SuperCA).

**Application Tests**

The following sections present the test results of each tested application.
Email Application: Outlook Express 6.0 SP1, OS: Windows 2000, Windows XP

Signature validation: First the PKCS #12 file is imported providing the local user with a key and certificate as well as the complete chain of its own PKI A. A signed email from user B of PKI B is still displayed as invalid because of the untrusted root certificate of PKI B. The SuperCA certificate and both replacement certificates for the original roots of PKI A and PKI B are imported using the PKCS #7 file. The new SuperCA certificate is presented to the user for acknowledgment. After installation of the certificates a signed mail is displayed as trusted, the displayed certificate chain shows the full path up to the SuperCA certificate:

In another setting only the SuperCA certificate was imported. A previously signed email (containing only the certificates of user B, subCA B and root B) was still displayed as being untrusted. This is due to the missing link to the SuperCA certificate. When an email is received that contains the new certificate chain (user B, subCA B and the replacement of rootB), the certificate chain is complete and the email is shown as being validly signed. Additionally the replacement certificate of PKI B is imported automatically so the previously signed email can now be validated, too.

Encryption: Same as scenario 2.

Summary: Outlook Express validates the new certificate chain up to the SuperCA certificate though the other root certificates still exist. Obviously all existing paths are validated, so if the SuperCA certificate exists and is trusted, the validation of the signed emails from both linked PKIs works.

Email Application: Outlook XP Pro SP2, OS: Windows 2000, Windows XP

Signature validation: If only the certificate of the new SuperCA is imported (and trusted), Outlook still fails to verify the signature of a signed email from user B, when the mail only contains the old certificate chain (certificates of user B, subCA B and rootCA B). An email containing the full new certificate chain (user B, subCA B, newrootCA B) is validated successfully. But the contained intermediate certificates are not imported into the certificate manager, so even after successfully validating the new email, the replacement certificate of rootCA B (newrootCA B) is not known to the certificate manager. Either all new certificates are locally imported (using a PKCS #7 file containing SuperCA, newrootCA A, newrootCA B) or a signed email with the full new certificate chain has to be used to add or update the contact entry of the sender. This is done using the context menu when selecting the “from” entry of the signed message (see scenario 1, Outlook XP). After this procedure all necessary certificates are known to the certificate manager and thus the email can be validated. The certificate chain is displayed correctly even if the email itself did not contain the new chain.
To establish a new PKI using a SuperCA certificate it seems to be most practical to provide a PKCS7 file containing the SuperCA certificate and the new intermediate certificates for the former root CAs.

**Encryption:** For the encryption adding the sender of a signed email to the contacts is necessary to import his user certificate. Again, after encrypting the email, Outlook states that the email is encrypted for the local user (sender) only.

**Summary:** Outlook is able to validate the certificate chains to the newly created SuperCA certificate. Unfortunately the import of the intermediate CA certificates (especially newrootCA A & newrootCA B) requires manual interaction even if the SuperCA certificate itself is already imported and trusted. The easiest way could be to provide all the necessary new certificates in a PKCS7 file.

**Email Application:** Mozilla 1.7.2, OS: Windows 2000, Windows XP, Linux

**Signature validation:** For the complete establishment of the new SuperCA, three certificates have to be imported (the SuperCA certificate itself and the replacement certificates for rootCA A and rootCA B). As tested in scenario 2, Mozilla is not able to import multiple certificates using a PKCS #7 file. So initially only the SuperCA certificate was imported into the certificate manager and trusted to identify mail users. A signed email containing the old certificate chain of PKI B (user B, subCA B and rootCA B) cannot be validated, because the certificate newrootB is missing to complete the chain up to the SuperCA.

If a signed email already contains the complete new certificate chain, the signature can be validated.
Because the SuperCA certificate is already a trusted CA certificate, the intermediate certificate newrootB is imported automatically from the mail. This means that after validating the signed mail with the complete new certificate chain, the mail containing only the old certificate chain can be validated, too.

So obviously the concept of using a SuperCA to link two PKIs works for Mozilla, though it requires the signed emails that should be validated to contain the new certificate chain up to the SuperCA.

**Encryption:** A user certificate needs to be installed to enable Mozilla to encrypt emails, because it always encrypts to the sender of the email, too. After importing an own key and certificate (using PKCS #12) and configuring the certificate as "own certificate" (in *Edit -> Mail and Newsgroups Account Settings -> UserAccount -> Security*) encryption works as expected.

**Summary:** Mozilla is able to support a SuperCA, the newly issued certificates for rootCA A and rootCA B do not conflict with the old certificates of these rootCAs. An import of a PKCS #7 file containing all necessary certificates would ease the usage of the SuperCA method, right now Mozilla needs either to import each of these certificates manually or the intermediate certificates (newrootCA A and newrootCA B) have to be provided by signed emails.

**Email Application: Thunderbird 1.0, OS: Windows 2000, Windows XP, Linux**

In scenario 3 Thunderbird behaves exactly the same as Mozilla 1.7.2 in all three operating systems: The import of the SuperCA certificate and both replacement certificates for rootCA A and rootCA B is not possible using a single PKCS #7 file, after importing of the single
SuperCA certificate, signature validation works for emails containing the full new certificate chain. Alternatively the replacement certificates (newrootCA A and newrootCA B) can be imported manually. After any one of these steps, signature validation works even for emails that contain only the old certificate chain (up to oldrootCA B). For further details see the description of Mozilla Mail above.

**Email Application: Kmail 1.7.1 (KDE 3.3.0), OS: Linux**

**Signature validation:** Kmail was configured as shown in scenario 1, the first signed email is displayed as not verified. Then the SuperCA certificate and the issued rootCA B certificate are imported using the certificate manager kleopatra. The email signature still remains not verified because of the untrusted SuperCA certificate. Surprisingly it remains untrusted after the SuperCA certificate is marked manually as trusted. After some tests it seems that **gpgsm** does not check all possible paths from the client certificate to the root certificate. Even more surprisingly the removal of the root certificate of PKI B lead to a successful signature validation, even if the root certificate is reimported into the keyring immediately. The only visible difference was the order of the certificates in the keyring, so **gpgsm** seems to search for issuer certificates in a "first match" search (still to be validated by the GPG developers). These circumstances make a distribution of additional root certificates difficult, sending a signed email with the new certificates and marking the SuperCA is not sufficient, the keyring of stored certificates has to be manipulated. In a closed environment this may be done in an automated way, for loosely coupled organisations this does not seem feasible.

**Encryption:** Encryption works as described in scenario 01, when the SuperCA certificate is configured to be trusted.

**Summary:** The linking of different PKIs using a SuperCA certificate suffers from the incomplete certificate chain search of **gpgsm**. It seems that the order of certificates in the keyring (which is a result of the order of importing the certificates) is relevant for a successful validation. So additional manipulation of the users keyring and configuring the SuperCA certificate as trusted (in the file ~/.gnupg/trustlist.txt) is necessary. In order to send encrypted emails either a signed email from the designated recipient can provide his complete certificate chain (including his end user certificate) or LDAP solutions need to be applied.

**Web Application: Internet Explorer 6.0 SP1, OS: Windows 2000, Windows XP**

**Server Authentication:** For a user that already installed all the necessary certificates and keys of his own PKI (PKI A), the import of a SuperCA certificate should provide the link to other PKIs. The SuperCA certificate can be imported by selecting the single certificate from the single_certs directory, Internet Explorer presents the certificate to the user for verification. After this import the additional replacement certificates for the former rootCAs are not known to the browser. Internet Explorer is already able to establish a trusted connection to the testserver, if the server provides the full certificate chain up to the SuperCA certificate (certificates of server B, subCA B, newrootCA B).
If the server does not provide the path to the SuperCA the usual warnings about untrusted authority certificates are displayed. The missing certificates to establish a trust path are the replacement certificates. An easy solution for Internet Explorer is to provide these together with the SuperCA certificate in a PKCS #7 file (see scenario 2). Internet Explorer asks the user to verify only the SuperCA certificate. Afterwards all necessary certificates are known, so connecting to the testserver does not display any warnings even if the server provides only the old certificate chain which leads up to the root certificate of PKI B. Internet Explorer builds the complete chain up to the SuperCA certificate, even if the server provides only a part of this chain:

Summary: Internet Explorer supports building trust paths between PKIs using the establishment of a new SuperCA. The replacement certificates for rootCA A and rootCA B do not interfere with the other root certificates. An easy way to distribute the necessary certificates is a PKCS #7 file containing not only the SuperCA certificate but also the certificates of newrootCA A and newrootCA B.

Web Application: Mozilla 1.7.2, OS: Windows 2000, Windows XP, Linux

Server Authentication: The import of the SuperCA certificates is the same as in Mozilla Mail (see above), PKCS #7 is not supported for this usage. After installation of the single SuperCA certificate, server authentication works with a server providing the full new certificate chain (especially the replacement-certificate of rootCA B). If the replacement certificates for the rootCAs are installed in the browser, server authentication works even for servers providing the old certificate chain.

Summary: Mozilla is able to support a SuperCA, the newly issued certificates for rootCA A and rootCA B do not conflict with the old certificates of these rootCAs. An import of a PKCS #7 file containing all necessary certificates would ease the usage of the SuperCA method, right now Mozilla needs to import each of these certificates manually.

Web Application: Firefox 1.0, OS: Windows 2000, Windows XP, Linux

In scenario 3 Firefox behaves exactly the same as Mozilla 1.7.2 in all three operating systems: The import of the SuperCA certificate and both replacement certificates for rootCA A and rootCA B is not possible using a single PKCS #7 file, after importing of the single SuperCA certificate, server authentication works, if the server provides the complete new certificate chain (including the replacement certificate of rootCA B). If the replacement certificates are imported into the browser, server authentication works even for servers providing only the old
certificate-chain up to rootCA B. For further details see the description of the Mozilla web browser above.

**Web Application: Opera 7.54, OS: Windows 2000, Windows XP, Linux**

**Server Authentication:** First a PKCS #12 file containing all the necessary certificates and keys for a user of PKI A was imported (see Opera in scenario 1 for further details on PKCS #12 import). As expected after the import connections to a server of PKI B are still not trusted by default because no trust-path exists for PKI B. In a first step only the SuperCA certificate was installed in the browser (by importing it from the `single_certs` directory into the Opera certificate manager). The trust settings of the certificate were set to "allow connections", the warnings were switched off. After this procedure a connection to a testserver can be established without warnings, if the server provides the full certificate chain (including the replacement certificate `newrootCA B` which leads to the SuperCA certificate).

If the server does only provide the original certificate chain of PKI B (including the certificates of server B, subCA B and rootCA B), the replacement certificate `newrootCA B` is missing to establish a path to the trusted SuperCA certificate. If `newrootCA B` is imported into the browser (and warnings for this certificate are switched off), trusted connections to servers providing only the old certificate-chain of PKI B can be established without warnings.

**Summary:** Opera supports the method of using a SuperCA to link different PKIs. The replacement certificates for the former rootCAs do not interfere with their predecessors, Opera builds the correct trust path to the trusted SuperCA certificate, even if the servers provide only a certificate chain with an old root certificate.

**Web Application: Konqueror 3.3 (KDE 3.3.0), OS: Linux**

**Server Authentication:** The first connection attempt leads to the same results as the test in scenario 1. The import of the prepared PKCS #7 file does not work, obviously PKCS #7 is not supported by the KDE environment. This behaviour is independent from the suffix of the filename, if the file is imported with the suffix "*.der", the import fails (as shown in scenario 2).

After the manual import of the SuperCA certificate, the certificate chain from the server can already be validated if the server provides the complete chain with the certificate for rootCA B from the SuperCA and the remaining chain of PKI B. So the usage of a SuperCA certificate does work.

The opposite case where the client imports not only the SuperCA certificate but also the issued replacement certificate for rootCA B works, too. This way the certificate chain provided by the server does not have to be changed. In this case, the client has to link two parts of the complete chain, the chain of server certificate, subCA certificate and rootCA certificate provided by the server and the locally available chain of the SuperCA certificate and the issued certificate for the rootCA of PKI B. Konqueror displays only the certificate
chain provided by the server, the rootCA certificate of PKI B is marked as "not relevant", because the trustpath only needs the rootCA certificate issued by the SuperCA.

As already tested in scenario 1, the manual import of single certificates works as expected, in larger setups CA certificates can be provided in the according KDE files (see Konqueror tests scenario 1).

**Summary:** Obviously Konqueror supports the certificate chain validation with multiple possible paths. So the SuperCA certificate and the issued root CA certificate is accepted as a new path to a trusted root. The unnecessary certificates still provided by the webserver are correctly marked as "not relevant".
**Test Scenario 4: Cross Certificate, Root signs Root**

In this test scenario, a federation of two PKIs is built using a cross certificate. The cross certificate is issued by the root CA of PKI A, which signs the key of the root CA of PKI B. Only the cross certificate has to be distributed to the clients of PKI A for validation of signatures from PKI B.

**Email Test Preparations**

For the tests two PKIs are created (PKI A and PKI B), each consisting of a root CA, a subCA and a user certificate. From both PKIs a complete set of a users certificates (containing key, certificate and certificate chain) is exported as PKCS #12 file. A signed email is created using the actual certificate chain of PKI B. Then a cross certificate is issued from the root CA of PKI A for the root CA of B. The cross certificate is exported to a DER file. Again a signed email is created, now containing the complete certificate chain of PKI B plus the cross certificate. The script `build_scenario_04.pl` generates two PKCS #12 files, a DER file and two signed emails:

- `scenario-04.PKI-B.p12`: Contains a full set of data for PKI B (user-key, user-certificate, subCA certificate and rootCA certificate). Importing this file enables the user B to generate signed emails (that can be tested by user A).
- `scenario-04.PKI-A.p12`: Contains a full set of data for PKI A (user-key, user-certificate, subCA certificate and rootCA certificate). This file gives user A the trust basis for the cross certificate, which is imported later on.
- `cross-cert.der`: Contains the cross certificate in DER format.
- `scenario-04.signedmail`: Contains an S/MIME email signed by user B. In the PKCS #7 attachment is the signature of the mail and the complete certificate chain of PKI B. The file is already in MBOX format for easy testing in a local environment.
- `scenario-04.signedmail-with-cross`: Same as previous mail file, but the cross certificate was included in the email. This file can be used for further testing if the trust path can not be validated after importing the cross certificate.

The test can be conducted in two different ways. The first one is to import the file `scenario-04.PKI-A.p12` into the client and then point the application to use the file `scenario-04.signedmail-with-cross` as a mail folder. The signature of the test email created by user B from PKI B should now be valid, because the full trust path including the cross certificate is provided in the email.

If clients are not able to produce emails including the cross-certificate in the chain, another setting can be used: The cross certificate is imported into the client. The email now does not need to contain the cross certificate any more. So the further test can be conducted using the email `scenario-04.signedmail`. After this, the signature should be valid, if the application interprets the cross certificate correctly and builds a PKI federation of A and B.

More extensive testing can be conducted using a full email setting including a sending client, an email environment and a receiving client. On the sending side, the file `scenario-
04.PKI-B.p12 should be imported into the client application. Signed emails from PKI B can now be produced using the application. On the receiving side, scenario-04.PKI-A.p12 must be installed. Now the cross-certificate can be installed in the receiving client (or in the sending client to check if certificate chains containing the cross-certificate can be included in the signed emails).

**Web Test Preparations**

For the web tests, the script `build_scenario_04.pl` creates the usual files (server certificate, key and chain for PKI B, see Scenario 1), which must be installed on the test web server. First, the users certificates of PKI A (including the CA certificates) are installed on the client. An SSL connection to the test server should produce the usual browser dialogs related to untrusted certificates.

For a first test on the server side, an additional file `scenario-04.PKI-B-serverchain-with-cross.pem` is created, which contains a certificate chain with the certificates of the server CA and the root CA of PKI B and the cross certificate that links both PKIs. So if on the client side PKI A is already trusted to sign webservers, a full validation path is given to the client on connection establishment which can lead to the successful validation of the server certificate. If this still does not work, the next test would install the cross-certificate in the browser by importing `cross-cert.der`. Now the server chain can be reduced to contain only the server certificate and the certificates of subCA B and rootCA B. After these changes an SSL connection should be established without the usual browser dialogs concerning untrusted certificates. Additional configuration of trust settings may be necessary to allow the root CA A (and eventually the CAs on the path to the subCA of PKI B) to certify webservers.

**Application Tests**

The following sections present the test results of each tested application.

**Email Application: Outlook Express 6.0 SP1, OS: Windows 2000, Windows XP**

**Signature validation:** On importing the PKCS #12 file of PKI A, Outlook Express asks for the trust of the contained root certificate. A signed email containing only the certificate chain up to the root of PKI B still cannot be validated because of the missing cross certificate. Instead of the email content Outlook Express first displays a message that explains the reason of the invalid signature, afterwards the email content is shown with an icon of the invalid signature.
If the email already contains the cross certificate, it is imported automatically (because it is issued by PKI A which is already trusted). Otherwise the cross certificate has to be imported manually using the certificate manager. After each of these steps both emails are displayed as trustworthily signed, the full certificate chain can be displayed by clicking on the valid signature icon:

![Certificate Manager]

So using the cross certificate in a signed mail is a means to link both PKIs without user interaction.

**Encryption**: Same as scenario 2.

**Summary**: Outlook Express is able to handle cross certificates issued by the root of one PKI to link to the root of another PKI. No user interaction is required if the cross certificate is received in a signed email.

**Email Application**: Outlook XP Pro SP2, OS: Windows 2000, Windows XP

**Signature validation**: For the cross certificate to work, first the certificates of the local users PKI (PKI A) have to be imported. This is done using a PKCS #12 file as described previously (see scenario 1, Outlook Express). For the establishment of trust into PKI B, the cross-certificate "root A signs root B" is required. If a signed email from user B does not contain the cross-certificate, the signature cannot be validated. Either a signed mail containing the cross-certificate is added to the contacts (as described in scenario 1, Outlook XP) or the cross-certificate is imported manually into the Windows certificate manager. If rootCA A is trusted, signed emails containing the full chain including the cross certificate can be validated immediately.

![Email Example]

If the cross certificate is imported into the certificate manager, even emails containing the old certificate chain (user B, subCA B and rootCA B) are validated using the cross certificate:
Encryption: For the encryption adding the sender of a signed email to the contacts is necessary to import his user certificate. Again, after encrypting the email, Outlook states that the email is encrypted for the local user (sender) only.

Summary: With Outlook XP, cross certificates still need to be manually imported (either by creating a contact from a signed email or by manual import into the Windows certificate manager). This way integration of new PKIs is not as transparent as it is with Outlook Express. The validation of the certificate chains using the cross certificate works flawless.

Email Application: Mozilla 1.7.2, OS: Windows 2000, Windows XP, Linux

Signature validation: First a PKCS #12 file containing the key and certificates of user A is imported. The root certificate of PKI A is configured to be trusted (see scenario 1, Mozilla Mail for more information). A signed email containing only the certificate chain of PKI B still cannot be validated because the cross certificate from PKI A to PKI B is still missing.

Accordingly the trust chain can only be built up to the root certificate of PKI B, which is untrusted. If a signed email already contains the complete new certificate chain (especially the cross-certificate), the signature can be validated:

Because the rootCA A is already a trusted authority, the intermediate cross-certificate is imported automatically from the mail. This means that after validating the signed mail with
the complete new certificate chain, the mail containing only the old certificate chain can be validated, too:

So obviously the certificate chain from the mail of user B is built up to the root certificate of PKI A using the cross certificate, even if the email does not contain the cross certificate:

The concept of using a cross-certificate from rootCA A to rootCA B works for Mozilla. It even does not require a manual interaction of the local user if the cross certificate is distributed in a signed mail.

**Encryption:** Because with the PKCS #12 file a certificate and key for the local user A was already installed, encryption works after setting the users own configuration for the mail account used (in Edit -> Mail & Newsgroups Account Settings -> UserAccount -> Security).

**Summary:** Usage of a cross certificate to link both PKIs does not even require a manual interaction in the clients application. The cross-certificate links both PKIs unnoticed by the user when it is distributed in the signed emails. Otherwise it has to be imported manually like a different root-certificate, though it does not require additional trust settings.
Email Application: Thunderbird 1.0, OS: Windows 2000, Windows XP, Linux

In scenario 4 Thunderbird behaves exactly the same as Mozilla 1.7.2 in all three operating systems: When the root-certificate of PKI A is trusted, a signed email from PKI B can be validated immediately, if the cross certificate exists. The cross certificate can be imported manually if provided in a signed email. After the import of the cross certificate, even emails containing only the old certificate chain (up to rootCA B) can be validated. For further details see the description of Mozilla above.

Email Application: Kmail 1.7.1 (KDE 3.3.0), OS: Linux

Signature validation: Kmail was configured as shown in scenario 1, the first signed email is displayed as not verified. Then the root certificate of PKI A and the cross certificate issued to the root CA of PKI B are imported using kleopatra. Even after manually marking the root certificate of PKI A as trusted, the client certificate of PKI B cannot be validated. The behaviour is the same as in scenario 3, the removal of the root certificate of PKI B leads to a successful signature validation, gpgsm seems only to validate the first path found leading from a client to a known root certificate. See scenario 3 for further information.

Encryption: Encryption works as described in scenario 01, when the root certificate of PKI A is correctly configured as trusted.

Summary: The linking of different PKIs using a cross certificate from the root of PKI A to the root of PKI B seems to suffer from an incomplete certificate chain search of gpgsm. It seems that the order of certificates in the keyring (which is a result of the order of importing the certificates) is relevant for a successful validation. So additional manipulation of the users keyring and configuring the root certificate of PKI A as trusted (in the file ~/.gnupg/trustlist.txt) is necessary. In order to send encrypted emails either a signed email from the designated recipient can provide his complete certificate chain (including his end user certificate) or LDAP solutions need to be applied.

Web Application: Internet Explorer 6.0 SP1, OS: Windows 2000, Windows XP

Server Authentication: For a user that already installed all the necessary certificates and keys of his own PKI (PKI A), connections to a webserver of PKI B providing the full certificate chain (including the cross-certificate) do not produce any warnings. The usage of the cross-certificate usually will go unnoticed by the user. If the server does not provide the cross-certificate, no trust is given to PKI B, so warnings are still displayed.

If the cross-certificate is imported using the Windows certificate manager, the user does not have to verify the imported certificate (because it is part of the already trusted PKI A). After this procedure, connections to servers of PKI B are "alert-free", even if they only provide the normal certificate chain of PKI B.

In both cases the certificate chain from the server of PKI B to the rootCA of PKI A is displayed correctly:

(c) 2004-2005 SURFnet and DFN-CERT Services
Summary: Internet Explorer supports the usage of cross certificates. If servers provide the cross-certificate in the transferred certificate chain, no user interaction is necessary. Otherwise the cross-certificate has to be imported using the Windows certificate manager.

Web Application: Mozilla 1.7.2, OS: Windows 2000, Windows XP, Linux

Server Authentication: The import of the certificates is the same as in Mozilla Mail (see above): After installation of the own PKIs certificates from a PKCS #12 file (user-cert, user-key, subCA A, rootCA A), server authentication works immediately for servers providing the full certificate chain (including the cross-certificate). If the cross certificate is installed in the browser, server authentication works even for servers providing the old certificate chain.

Summary: Usage of a cross-certificate to link both PKIs does not even require a manual interaction in the clients application. The cross certificate links both PKIs unnoticed by the user when it is sent in the certificate chain of the server. Otherwise it has to be imported manually like a different root certificate, though it does not require additional trust settings.

Web Application: Firefox 1.0, OS: Windows 2000, Windows XP, Linux

In scenario 4 Firefox behaves exactly the same as Mozilla 1.7.2 in all three operating systems: When the root-certificate of PKI A is trusted, server authentication of a server of PKI B succeeds immediately, if the server provides the full certificate chain (including the cross certificate). If the cross certificate is imported into the browser, server authentication works even for servers providing only the old certificate chain up to rootCA B. For further details see the description of the Mozilla WWW-browser above.

Web Application: Opera 7.54, OS: Windows 2000, Windows XP, Linux

Server Authentication: First a PKCS #12 file containing all the necessary certificates and keys for a user of PKI A was imported (see Opera in scenario 1 for further details on PKCS #12 import). The trust settings of the root certificate of PKI A have to be edited to switch off warnings. After this procedure a connection to a test server can be established without warnings, if the server provides the full certificate chain (including the cross certificate from rootCA A to rootCA B).

If the server does only provide the original certificate chain of PKI B (including the certificates of server B, subCA B and rootCA B), the cross certificate is missing to establish a...
path to the trusted root-certificate of PKI A. If the cross certificate is imported into the browser (and warnings for this certificate are switched off), trusted connections to servers providing only the old certificate chain of PKI B can be established without warnings.

If the warnings for the cross certificate are still switched on, Opera displays a very odd message:

![Certificate warning](image)

It seems that the name of the certificate to be warned about is empty because it is not provided in the certificate chain of the server (the cross certificate is already installed in the browser and not provided by the server).

**Summary:** Opera supports the linking of a foreign rootCA to the own PKI by issuing a cross certificate. Connections to servers providing the full certificate chain (including the cross certificate) work without user interaction. When the cross certificate is installed in the browser, Opera builds the correct trust path to the own trusted root certificate even if the foreign servers provide only a certificate chain up to their own root certificate.

**Web Application: Konqueror 3.3 (KDE 3.3.0), OS: Linux**

**Server Authentication:** The first connection attempt leads to the same results as the test in scenario 1. The root CA certificate of PKI B is imported from the provided DER-file. After this import the certificate chain from the server can already be validated if the server provides the complete chain including the cross certificate and the remaining chain of PKI B. So the usage of a root-signs-root cross certificate does work. The included root certificate of PKI B is marked as "not relevant".

The opposite case where the client imports not only the rootCA certificate of PKI A, but also the issued cross certificate to PKI B works as well. This way the certificate chain provided by the server does not have to be changed. In this case, the client has to link two parts of the complete chain, the chain of server certificate, subCA certificate and rootCA certificate of PKI B provided by the server and the locally available chain of rootCA A and the cross certificate to rootCA B. Konqueror displays only the certificate chain provided by the server,
the rootCA certificate of PKI B is marked as "not relevant", because the trust path only needs
the cross certificate issued by the rootCA A.

As already tested in scenario 1, the manual import of single certificates works as expected, in
larger installations CA certificates can be provided in the according KDE files (see konqueror
tests scenario 1).

**Summary:** Obviously Konqueror supports the certificate chain validation with multiple
possible paths. So the cross certificate "root-signs-root" is accepted as a new path to a trusted
root. The unnecessary certificates still provided by the web server are correctly marked as
"not relevant".
**Test Scenario 5: Cross Certificate, Root signs SubCA**

This test scenario is very similar to scenario 4, the only difference being the cross certificate. This time it is still issued by the root CA of PKI A, but it signs not the rootCA but the subCA of PKI B. So the PKI federation includes the complete PKI A but only a part of PKI B.

**Email Test Preparations**

The script `build_scenario_05.pl` works identical to the script for scenario 4, only the created cross certificate differs. It is expected that the results of the tests are quite similar to those of scenario 4.

**Web Test Preparations**

For the web tests, the script `build_scenario_05.pl` creates the same additional files for the SSL tests as scenario 4. The tests are conducted in the same way, the only difference being the CAs along the certificate chain from PKI A to PKI B (this may be relevant if the trust settings of each CA along the path has to be edited manually).

**Application Tests**

The following sections present the test results of each tested application.

---

**Email Application: Outlook Express 6.0 SP1, OS: Windows 2000, Windows XP**

**Signature validation:** The process works exactly as in scenario 4, the results are the same. The only difference is of course the certificate chain displayed, the cross certificate linking both PKIs is now issued to the subCA of PKI B.

![Certificate Chain](image)

**Encryption:** Same results as in scenario 2.

**Summary:** Outlook Express validates certificate chains including cross certificates. These do not have to link root CAs, it is also possible to issue cross certificates for subCAs. No user interaction is required if the cross certificate is provided in a signed email.
Email Application: Outlook XP Pro SP2, OS: Windows 2000, Windows XP

Signature validation: The setting using the cross-certificate from rootCA A to subCA B is indeed very similar to scenario 4. Everything works exactly as already described there, the only difference of course being the displayed certificate chain of a trusted email signature:

![Certificate Chain Image]

Encryption: See scenario 4, Outlook XP.

Summary: With Outlook XP, cross-certificates still need to be manually imported (either by creating a contact from a signed email or by manual import into the Windows certificate manager). This way integration of new PKIs is not as transparent as it is with Outlook Express. The validation of the trust chains using the cross-certificate works flawless.

Email Application: Mozilla 1.7.2, OS: Windows 2000, Windows XP, Linux

Signature validation: The test with the cross certificate issued from rootCA A to the subCA B works the same as the cross certificate linking both rootCAs (see scenario 4, Mozilla Mail). The only difference is the displayed certificate chain, which now leads from the usercert B to subCA B and then directly to the rootCA A. This way additional subCAs of PKI B would not be trusted.

The concept of using a cross certificate from rootCA A to subCA B works for Mozilla. It even does not require a manual interaction of the local user if the cross certificate is distributed in a signed mail.

Encryption: Same as in scenario 4.

Summary: Usage of a cross certificate to link both PKIs does not even require a manual interaction in the clients application. The cross certificate links both PKIs unnoticed by the user when it is distributed in the signed emails. Otherwise it has to be imported manually like a different root certificate, though it does not require additional trust settings.

Email Application: Thunderbird 1.0, OS: Windows 2000, Windows XP, Linux

In scenario 5 Thunderbird behaves exactly the same as Mozilla 1.7.2 in all three operating systems, the results do not differ from the results from scenario 4 (the only difference is the provided certificate chain itself). For further details see the description of Mozilla above.

Email Application: Kmail 1.7.1 (KDE 3.3.0), OS: Linux

Signature validation: The results of the tests are nearly the same as in scenario 4. This time
the cross certificate has to be stored in the keyring prior to the import of the subCA certificate of PKI B, otherwise the email cannot be validated. See scenario 3 for further information.

**Encryption:** Encryption works as described in scenario 1 when the root certificate of PKI A is correctly configured as trusted.

**Summary:** The linking of different PKIs using a cross certificate from the root of PKI A to the root of PKI B seems to suffer from an incomplete certificate chain search of gpgsm. It seems that the order of certificates in the keyring (which is a result of the order of importing the certificates) is relevant for a successful validation. So additional manipulation of the users keyring and configuring the root certificate of PKI A as trusted (in the file ~/.gnupg/trustlist.txt) is necessary. In order to send encrypted emails either a signed email from the designated recipient can provide his complete certificate chain (including his end user certificate) or LDAP solutions need to be applied.

**Web Application: Internet Explorer 6.0 SP1, OS: Windows 2000, Windows XP**

**Server Authentication:** The results are exactly the same as for scenario 4. The only difference is of course the used certificate-chain, which is displayed correctly by Internet Explorer:

![Internet Explorer certificate chain](image)

**Summary:** Internet Explorer supports the usage of cross-certificates. If servers provide the cross-certificate in the transferred certificate chain, no user interaction is necessary. Otherwise the cross-certificate has to be imported using the Windows certificate manager.

**Web Application: Mozilla 1.7.2, OS: Windows 2000, Windows XP, Linux**

**Server Authentication:** The test with the cross certificate issued from rootCA A to the subCA B works the same as the cross certificate linking both rootCAs (see scenario 4, Mozilla web browser). The only difference is the displayed certificate chain, which now leads from the usercert B to subCA B and then directly to the rootCA A. This way additional subCAs of PKI B would not be trusted.

After installation of the own PKIs certificates from a PKCS #12 file (usercert, userkey, subCA A, rootCA A), server authentication works immediately for servers providing the full certificate chain (including the cross certificate). If the cross certificate is installed in the browser, server authentication works even for servers providing the old certificate chain.

**Summary:** Usage of a cross-certificate to link both PKIs does not even require a manual interaction in the clients application. The cross certificate links both PKIs unnoticed by the
user when it is sent in the certificate chain of the server. Otherwise it has to be imported manually like a different root certificate, though it does not require additional trust settings.

**Web Application: Firefox 1.0, OS: Windows 2000, Windows XP, Linux**

In scenario 5 Firefox behaves exactly the same as Mozilla 1.7.2 in all three operating systems, the results do not differ from the results from scenario 4 (the only difference is the provided certificate chain itself). For further details see the description of the Mozilla web browser above.

**Web Application: Opera 7.54, OS: Windows 2000, Windows XP, Linux**

**Server Authentication:** The test is conducted like scenario 4, the PKCS #12 file of PKI A is imported into the browser and the warnings for the contained CA certificates are switched off. After this procedure a connection to a test server can be established without warnings, if the server provides the full certificate chain (including the cross certificate from rootCA A to subCA B).

If the server provides only the original certificate chain of PKI B (including the certificates of server B, subCA B and rootCA B), the cross-certificate is missing to establish a path to the trusted root-certificate of PKI A. But after the import of the cross certificate into the browser trying to connect to servers providing only the old certificate chain of PKI B produces the following alert:

![Certificate signe not found alert](image)

Opera is not able to build the link between the provided certificate chain from the server and the chain of certificates already installed locally. It may be that Opera only checks for locally installed certificates at the end of the provided certificate chain. At least it is possible to use the imported cross certificate if the server provides no chain at all but only its own server certificate. Opera then finds the issuer of the server certificate (which in this case is the cross certificate itself). So in scenario 5 it is not possible to import the cross-certificate into the
browsers and not changing the servers of the foreign PKI (which could cause new problems in the foreign PKI). For Opera alone it may be sufficient to always provide a certificate chain containing the chain of the own PKI as well as all existing cross certificates, but this could cause problems with other applications.

**Summary:** Obviously the linking from the own rootCA to a subCA of a foreign PKI is not fully supported by Opera. When the cross certificate is not installed in the browser, the servers may provide all necessary certificates (including the cross certificate). This setting could prove not to work out for all applications (especially for server software that does not offer the possibility of changing the provided certificate chain).

**Web Application: Konqueror 3.3 (KDE 3.3.0), OS: Linux**

**Server Authentication:** The first connection attempt leads to the same results as the test in scenario 1. The root CA certificate of PKI B is imported from the provided DER-file. After this import the certificate chain from the server can already be validated if the server provides the complete chain including the cross certificate and the remaining chain of PKI B. So the usage of a root-signs-subCA cross certificate does work. The included root certificate of PKI B is marked as "not relevant".

The opposite case where the client imports not only the rootCA certificate of PKI A, but also the issued cross certificate to PKI B while the server does provide the certificates rootCA B, subCA B und server B does not work. It seems that Konqueror does not validate all the possible paths up to trusted root certificates. If the server certificate chain only contains the cross certificate, only a single path is possible. Consequently Konqueror accepts the certificates. So in the usual case of distributing the cross certificate to the clients, the server configuration has to be changed as well.

As already tested in scenario 1, the manual import of single certificates works as expected, in larger installations CA certificates can be provided in the according KDE files (see Konqueror tests scenario 1).

**Summary:** Obviously Konqueror is not always able to validate certificate chains with multiple possible paths. So the cross certificate "root-signs-subCA" is accepted as a new path to a trusted root only if not multiple paths are provided. It depends on the web server configuration if the cross certificate is accepted.
Test Scenario 6: Cross Certificate, SubCA signs SubCA

This time the cross certificate is used to build a federation of the subCAs of PKI A and PKI B, so the subCA A issues a cross certificate for the subCA of PKI B. The test setting is almost identical to the scenarios 4 and 5.

Email Test Preparations

The script `build_scenario_06.pl` works identical to the script for scenario 4 and 5, only the created cross certificate differs. It is expected that the results of the tests are quite similar to those of the scenarios 4 and 5.

Web Test Preparations

For the web tests, the script `build_scenario_06.pl` creates the same additional files for the SSL tests as scenarios 4 and 5. The tests are conducted in the same way, the only difference being the CAs along the certificate chain from PKI A to PKI B (this may be relevant if the trust settings of each CA along the path has to be edited manually).

Application Tests

The following sections present the test results of each tested application.

Email Application: Outlook Express 6.0 SP1, OS: Windows 2000, Windows XP

Signature validation: The process works exactly as in scenario 4 and 5, the results are the same. The only difference is of course the certificate chain displayed, the cross certificate linking both PKIs is now issued by the subCA of PKI A to the subCA of PKI B.

Encryption: Same results as in scenario 2.

Summary: Outlook Express validates certificate chains including cross certificates. These do not have to link root CAs, it is also possible to issue cross certificates between subCAs. No user interaction is required if the cross certificate is provided in a signed email.
Email Application: **Outlook XP Pro SP2, OS: Windows 2000, Windows XP**

**Signature validation:** The setting using the cross certificate from subCA A to subCA B is indeed very similar to scenario 4 and scenario 5. Everything works exactly as already described there, the only difference being the displayed certificate chain of a trusted email signature:

![Certificate Chain](image)

**Encryption:** See scenario 4 & 5, Outlook XP.

**Summary:** With Outlook XP, cross certificates still need to be manually imported (either by creating a contact from a signed email or by manual import into the Windows certificate manager). This way integration of new PKIs is not as transparent as it is with Outlook Express. The validation of the trust chains using the cross certificate works flawless.

---

Email Application: **Mozilla 1.7.2, OS: Windows 2000, Windows XP, Linux**

**Signature validation:** The test with the cross certificate issued from subCA A to the subCA B works the same as the cross certificate linking in scenario 4 & 5 (see scenario 4, Mozilla Mail). The only difference is the displayed certificate chain, which now leads from the usercert B to subCA B, then to subCA A and finally to the rootCA A.

The concept of using a cross certificate from subCA A to subCA B works for Mozilla. It even does not require a manual interaction of the local user if the cross certificate is distributed in a signed mail.

**Encryption:** Same as in scenario 4 & 5.

**Summary:** Usage of a cross-certificate to link both PKIs does not even require a manual interaction in the clients application. The cross certificate links both PKIs unnoticed by the user when it is distributed in the signed emails. Otherwise it has to be imported manually like a root certificate, though it does not require additional trust settings.

---

Email Application: **Thunderbird 1.0, OS: Windows 2000, Windows XP, Linux**

In scenario 6 Thunderbird behaves exactly the same as Mozilla 1.7.2 in all three operating systems, the results do not differ from the results from scenario 4 and scenario 5 (the only difference is the provided certificate chain itself). For further details see the description of Mozilla above.
**Email Application: Kmail 1.7.1 (KDE 3.3.0), OS: Linux**

**Signature validation:** The results of the tests are the same as in scenario 5. For successful validation of the signed email, the cross certificate has to be stored in the keyring prior to the import of the subCA certificate of PKI B. See scenario 3 for further information.

**Encryption:** Encryption works as described in scenario 1 when the root certificate of PKI A is correctly configured as trusted.

**Summary:** The linking of different PKIs using a cross certificate from the subCA of PKI A to the subCA of PKI B seems to suffer from an incomplete certificate chain search of gpgsm. It seems that the order of certificates in the keyring (which is a result of the order of importing the certificates) is relevant for a successful validation. So additional manipulation of the users keyring and configuring the root certificate of PKI A as trusted (in the file ~/.gnupg/trustlist.txt) is necessary. In order to send encrypted emails either a signed email from the designated recipient can provide his complete certificate chain (including his end user certificate) or LDAP solutions need to be applied.

**Web Application: Internet Explorer 6.0 SP1, OS: Windows 2000, Windows XP**

**Server Authentication:** The results are exactly the same as for scenario 4 and 5. The only difference is of course the used certificate-chain, which is displayed correctly by Internet Explorer:

![Certificate chain in Internet Explorer](image)

**Summary:** Internet Explorer supports the usage of cross-certificates. If servers provide the cross-certificate in the transferred certificate chain, no user interaction is necessary. Otherwise the cross-certificate has to be imported using the Windows certificate manager.

**Web Application: Mozilla 1.7.2, OS: Windows 2000, Windows XP, Linux**

**Server Authentication:** The test with the cross certificate issued from subCA A to the subCA B works the same as the previous scenarios with cross certificates (see scenario 4 & 5, Mozilla web browser). The only difference is the displayed certificate chain, which now leads from the usercert B to subCA B, then to subCA A and finally to the rootCA A.

After installation of the own PKIs certificates from a PKCS #12 file (user-cert, user-key, subCA A, rootCA A), server authentication works immediately for servers providing the full certificate chain (including the cross certificate). If the cross certificate is installed in the browser, server authentication works even for servers providing the old certificate chain.
Summary: Usage of a cross certificate to link both PKIs does not even require a manual interaction in the clients application. The cross certificate links both PKIs unnoticed by the user when it is sent in the certificate chain of the server. Otherwise it has to be imported manually like a root certificate, though it does not require additional trust settings.

Web Application: Firefox 1.0, OS: Windows 2000, Windows XP, Linux

In scenario 6 Firefox behaves exactly the same as Mozilla 1.7.2 in all three operating systems, the results do not differ from the results from scenario 4 and scenario 5 (the only difference is the provided certificate chain itself). For further details see the description of the Mozilla web browser above.

Web Application: Opera 7.54, OS: Windows 2000, Windows XP, Linux

Server Authentication: The test of scenario 6 resembles the results of scenario 5 in some ways. The import of the certificates of PKI A using a PKCS #12 file works as expected, after that manually changing the trust settings of the imported certificates is necessary to prevent warnings. After this procedure a connection to a test server of PKI B is established without warnings if the server provides the full certificate chain (including the cross certificate from subCA A to subCA B).

If the server provides only the original certificate chain of PKI B (including the certificates of server B, subCA B and rootCA B), the cross certificate is missing to establish a path to the trusted root-certificate of PKI A. But in this scenario the import of the cross certificate fails:

![Certificate Installation Failed](image)

It seems that the cross-certificate interferes in some way with the already installed certificate of subCA A. It was possible to import the cross-certificate after deleting the subCA certificate, and it was even possible to import the subCA certificate then again. So all necessary certificates were imported, which did not help anyway: Opera showed the same behaviour as described in scenario 5, a successful connection was only possible if the server provided only its own server certificate and no certificate chain at all. Opera then finds the supposed issuer of the server certificate (which in this case is the cross certificate itself). So in scenario 6 it is not possible to import the cross certificate into the browsers and not changing the servers of the foreign PKI (which could cause new problems in the foreign PKI). For Opera alone it may be sufficient to always provide a certificate chain containing the chain of the own PKI as well as all existing cross certificates, but this could cause problems with...
other applications.

**Summary:** Obviously the linking from the own subCA to a subCA of a foreign PKI is not fully supported by Opera. When the cross certificate is not installed in the browser, the servers may provide all necessary certificates (including the cross certificate). This setting could prove not to work out for all applications (especially for server software that does not offer the possibility of changing the provided certificate chain).

**Web Application: Konqueror 3.3 (KDE 3.3.0), OS: Linux**

**Server Authentication:** The results are the same as in scenario 5. After the import of the root certificate of PKI A, the certificate chain from the server can already be validated if the server provides the complete chain including the cross certificate, the subCA certificate of PKI A and the remaining chain of PKI B. So the usage of a subCA-signs-subCA cross certificate does work. The included subCA certificate of PKI B as well as the root certificate of PKI B are marked as "not relevant".

The opposite case where the client imports not only the rootCA certificate of PKI A, but also the subCA certificate of PKI A and the issued cross certificate does not work. Again Konqueror seems not to be able to follow multiple paths. Changes in the server configuration may circumvent this problem (see scenario 5).

As already tested in scenario 1, the manual import of single certificates works as expected, in larger installations CA certificates can be provided in the according KDE files (see Konqueror tests scenario 1).

**Summary:** Obviously Konqueror is not always able to validate certificate chains with multiple possible paths. So the cross certificate "subCA-signs-subCA" is accepted as a new path to a trusted root only if not multiple paths are provided. It depends on the web server configuration if the cross certificate is accepted.
Test Scenario 7: Root-Certificate-Exchange with Cross-Certificates

This scenario presents a common solution for the distribution of new root-certificates. Before an old root certificate expires, a new one is created. To establish a trust path, two additional cross-certificates are build. The first one is the key of the new CA (in the test case rootCA B), signed by the old one (new-with-old), the second is the key of the old CA (rootCA A) signed by the new one. By providing the new root certificate and both cross certificates, a trust path is established between both, so the new certificate can be validated by the client immediately.

If the shell modell is used for signature validation, the trust may only exist as long as the old root certificate is not expired.

At the time $T$, the test scripts create a PKI A which is valid from $T - 1$ year to $T + 1$ year, and a PKI B, valid from $T$ to $T + 2$ years. So at the actual time, both PKIs are valid. The cross-certificates are both valid from $T$ to $T + 1$ year. For further tests the system time can be set to $T + 1$ year (when the cross-certificates are expired) to see if PKI B is still considered to be valid without additional user interaction.

Email Test Preparations

The script `build_scenario_07.pl` produces the following files:

- `scenario-07.PKI-A.p12`: Contains a full set of data for PKI A (user-key, user-certificate, subCA certificate and rootCA certificate). Importing this file enables the user A to generate signed emails (that can be tested by user B).

- `scenario-07.signedmail`: Contains an S/MIME email signed by user B. In the PKCS #7 attachment is the signature of the mail and the complete certificate chain of PKI B (subCA certificate and rootCA certificate). The file is already in MBOX format for easy testing in a local environment.

- `scenario-07.signedmail-with-cross`: Contains an S/MIME email signed by user B. The complete certificate chain of PKI B is included, this time both cross certificates are included as well.

- `scenario-07.cross-old-with-new.der`: Contains the cross certificate with the key of the old rootCA (rootCA A), signed by the new rootCA (rootCA B).

- `scenario-07.cross-new-with-old.der`: Contains the cross certificate with the key of the new rootCA (rootCA B), signed by the old rootCA (rootCA A).

Web Test Preparations

For the web tests, the script `build_scenario_07.pl` creates the usual files for the server configuration (server key, server certificate, server certificate chain). Additionally a second server certificate chain (`scenario-07.PKI-B-serverchain-with-cross.pem`) is provided containing a certificate chain with subCA B, rootCA B, cross certificate old-with-new & cross certificate new-with-old. The tests are conducted in the same way, the only difference being the CAs along the certificate chain from PKI A to PKI B (this may be
relevant if the trust settings of each CA along the path has to be edited manually). With the second chain, a server certificate from the new PKI B should already be valid if the root certificate of PKI A exists in the browser. The first serverchain should work if at least the cross certificate new-with-old is imported.

**Application Tests**

The following sections present the test results of each tested application.

**Email Application: Outlook Express 6.0 SP1, OS: Windows 2000, Windows XP**

**Signature validation:** First a PKCS #12 file is imported for a user of PKI A (as described in scenario 1). A signed mail by user B of PKI B (which is the new certificate hierarchy) is still untrusted because of the missing link between both PKIs. Importing the missing certificates could be done using a PKCS #7 file (as described in scenario 2), in this test the single certificates are imported manually into the certificate manager. After importing the cross certificate "new-with-old" which provides the link from the old PKI to the new one, the signed emails from user B are displayed as validly signed. Everything still works if the certificate "old-with-new" is imported, too. This provides users of PKI B to verify emails signed by users of the old PKI A (and introduces a loop in the graph of certificate chains).

An additional test was to set the date to a future date, when the root certificate of PKI A and the crosscert "new-with-old" are expired, while PKI B is still valid. Outlook Express displays the signed emails still to be valid, it seems that the date of signature creation is tested to be in the valid range of the certificates:

Accordingly although the email signatures are valid, the certificate chain of the signed emails...
is displayed to be invalid at the current system time:

Encryption: Same results as in scenario 2.

Summary: Outlook Express validates certificate chains including cross certificates between successive PKIs. Even if the prior PKI (and accordingly the cross certificate "new-with-old") expires, the signature made before the expiry date are still considered to be valid. In the case of Outlook Express cross certificates between successive PKIs provide a means to establish trust in the new hierarchy.

Email Application: Outlook XP Pro SP2, OS: Windows 2000, Windows XP

Signature validation: The normal case of this scenario is that the local user has already imported all the necessary certificates of his own PKI A. This is achieved by importing the generated PKCS #12 file as described earlier (see scenario 1, Outlook Express). When the user receives an emails signed by a user of the new PKI B, the signature is still untrusted if the email does not contain the cross certificate "new-with-old" (in other words "A signs B"), which establishes the trust in the new PKI.

Emails containing the cross certificates are displayed as being trusted immediately:

If the cross certificates are imported manually (using the Windows certificate manager) or if a
signed email containing the full certificate chain including the cross certificates is added to the contacts database (as described in scenario 1, Outlook XP), emails containing only the certificate chain of PKI B are trusted as well.

A short view into the list of trusted authorities or intermediate CAs in the certificate manager reveals that the new root certificate (rootCA B) is not imported into the certificate storage in any case. This means that after the expiry of the old root certificate (rootCA A) and the cross-certificate "A signs B" (which with respect to the shell validity model cannot be valid longer than the issuing CA certificate of rootCA A), no valid trust anchor exists for the application. So the initial trust using the cross certificate "new-with-old" does not hold on after the expiry of the old PKI, the certificate of rootCA B has still to be installed as a trusted root certificate. Otherwise the certificates of PKI B are all not trusted:

![Certificate Manager Screen](image)

Surprisingly after setting the system time to a date when PKI A is expired but PKI B is still valid, even the signatures from PKI B which were generated at a time when PKI A was still valid, are not validated any more.

![Signature Validation](image)

It seems that Outlook XP does not check the time of signature creation, but checks the validity of the certificates at the actual time. This is very different from the behaviour of Outlook Express, which still validates the signatures, even if the certificate chains are already expired.

**Encryption:** Encryption works as usual, of course as long as the own certificates (PKI A) are valid.

**Summary:** Though Outlook XP is able to handle cross-certificates, even if they establish cycles in the trust graph (which is the case with the cross-certificates "new-with-old" and "old-with-new"), it is not possible to establish a continuing trust into the new PKI B, because the root-certificate of PKI B is not imported into the certificate manager. So scenario 7 can
only be an intermediate solution as long as PKI A is still valid.

**Email Application: Mozilla 1.7.2, OS: Windows 2000, Windows XP, Linux**

**Signature validation:** First the certificate chain of the actual PKI A are imported using a PKCS #12 file and the contained root-certificate of PKI A is configured to be trusted (see scenario 1). A signed email from PKI B is considered to be untrusted, because the cross-certificate that links both PKIs is still missing. But even if the signed email contains all the missing certificates (in this case the certificates of user B, subCA B, rootCA B and the cross certificates "new-with-old" and "old-with-new") the intermediate certificates are not imported and the trust chain is not established using the cross certificates:

![Certificate chain image](attachment:image.png)

To examine the reasons for this behaviour, the signed email was verified using OpenSSL. OpenSSL seems to complain about the certificate chain containing a root certificate ("contains issuer cert"). If the root certificate is removed from the chain, OpenSSL still refuses to validate the signature ("no issuer cert"). If the root certificate of PKI B and the cross-certificate "old-with-new" are removed from the chain, signature validation works. Unfortunately the same does not work with Mozilla: The contained certificates are still not imported, the signature is not validated. Only if the cross certificate "new-with-old" is imported manually, signed emails from PKI B can be validated.

A big surprise awaits the user if the second cross certificate "old-with-new" is imported: When selecting a signed mail, Mozilla appears to be stuck in an infinite loop immediately. It seems that Mozilla is following the cycle in the certificate chain created by the cross certificates. So linking of successive PKIs using two cross certificates does not work with Mozilla, the process proposed in the PKIX standard is not supported.

**Encryption:** Was not tested after the hang-up on the cross certificate validation.

**Summary:** The process of linking successive PKIs using two cross-certificates is not supported by Mozilla. Even worse Mozilla hangs in an infinite loop when trying to evaluate the certificate chain.

**Email Application: Thunderbird 1.0, OS: Windows 2000, Windows XP, Linux**

In scenario 7 Thunderbird behaves exactly the same as Mozilla 1.7.2 (in all three operating systems): When only the cross certificate "new-with-old" is provided, the signed emails from
PKI B can be validated successfully. Even after the expiry of PKI A (and the cross certificate), the signatures are still valid, the certificates are checked against the date of the signature creation. The import of the second cross certificate ("old-with-new") creates a cycle in the certificate chains, which seems to cause trouble with Thunderbird: All signatures are displayed as being invalid. If the user wants to examine the reason and selects "View Signature Certificate", Thunderbird hangs in an infinite loop.

**Email Application: Kmail 1.7.1 (KDE 3.3.0), OS: Linux**

**Signature validation:** For the main part, the results are the same as for the other cross certificate tests (see scenario 4, 5 & 6). For successful validation of the signed email, the cross certificate has to be stored in the keyring prior to the import of the rootCA certificate of PKI B. See scenario 3 for further information. An additional test is to set the system clock to a date after the expiry of the root certificate of PKI A (and thereby after the expiry of the cross certificate). kleopatra displays nearly all the certificates as expired, which is not exactly true, it is just the trust relationship (which came from PKI A) that is expired (see following image, orange writing indicates expired certificates, untrusted root certificates are marked violet).

![Kleopatra certificate view](image)

In contrary kmail still regards the signature as valid. This behaviour is as expected because all the necessary certificates were valid at the time of signature creation.

**Encryption:** Encryption works as described in scenario 1 when the root certificate of PKI A is correctly configured as trusted.

**Summary:** The linking of different PKIs using a cross certificate from the root of PKI A to the root of PKI B seems to suffer from an incomplete certificate chain search of gpgsm. It seems that the order of certificates in the keyring (which is a result of the order of importing the certificates) is relevant for a successful validation. So additional manipulation of the users keyring and configuring the root certificate of PKI A as trusted (in the file ~/.gnupg/trustlist.txt) is necessary. In order to send encrypted emails either a signed email from the designated recipient can provide his complete certificate chain (including his end user certificate) or LDAP solutions need to be applied.
Web Application: Internet Explorer 6.0 SP1, OS: Windows 2000, Windows XP

Server Authentication: When the certificates of the previous PKI (PKI A) are already installed, connections to servers of the new PKI (PKI B) can already be established without warnings, if the servers provide the cross-certificate "new-with-old" in the transferred certificate chain. If the additional certificates ("new-with-old" and "old-with-new" and optionally the root-certificate of PKI-B) are imported into the certificate manager, even servers providing the normal chain of PKI B only can be verified successfully:

After the expiry of PKI A (and the cross certificate "new-with-old", which is issued by the rootCA of PKI A and therefore not longer valid than the issuing certificate), connections to the servers of PKI B are no longer trusted. On display of the certificate chain, Internet Explorer shows the according certificates of PKI A to be expired:

The certificate of subCA B is still valid (it is not expired already) but there is no trust in the root certificate of PKI-A any more, so the server certificate is not trusted. So providing a cross certificate from PKI A to PKI B may be an intermediate solution to provide trust into the new PKI, but it is not a final solution. Obviously the root certificate B is not installed as a trusted authority, the cross certificate has no lasting effect after its expiry.

Summary: The usage of cross certificates to establish trust from a PKI already in use to a successive PKI works. It is only an intermediate solution because the root certificate of the new PKI is never imported into the browser, so there is no lasting trust after the root certificate of the old PKI expires.

Web Application: Mozilla 1.7.2, OS: Windows 2000, Windows XP, Linux

Server Authentication: First the certificate chain of the actual PKI A is imported using a PKCS #12 file and the contained root certificate of PKI A is configured to be trusted (see scenario 1). Authentication of the test server of the new PKI B fails, even if the server provides the complete certificate chain (including the cross certificate "new-with-old"). After
some testing the problem could be circumvented: If the server can provide a handcrafted
certificate chain in which the second cross-certificate ("old-with-new") is not included, the
server authentication works. Obviously Mozilla has some difficulties in following certificate
chains if these contain cycles (which is the case if both cross certificates are evaluated). The
same happens if the cross certificates are imported directly into the browser and the server
does only provide the certificate chain up to the root-certificate of PKI B: If only "new-with-
old" is imported, everything works flawless, if the cross certificate "old-with-new" is
imported additionally, authentication of the same server does not work any more.

Consequently (compared to the behaviour of Mozilla Mail) the browser hangs in an infinite
loop if the user tries to view the certificate details.

**Summary:** The process of linking successive PKIs using two cross certificates is not
supported by Mozilla. Even worse Mozilla hangs in an infinite loop when trying to display
the certificate chain.

**Web Application: Firefox 1.0, OS: Windows 2000, Windows XP, Linux**

In scenario 7 Firefox behaves exactly the same as Mozilla 1.7.2 in all three operating systems:
When only the cross-certificate "new-with-old" is provided in the certificate-chain of the
server, server authentication succeeds. On expiry of the certificates of PKI A (especially the
root-certificate and the cross-certificate "new-with-old"), authentication fails because there is
no valid trust anchor in the list of trusted authorities. If the second cross-certificate ("old-
with-new") is either provided in the certificate-chain of the server or imported into the
browser, server authentication fails (even before the expiry of PKI A). Firefox hangs in an
infinite loop when the user tries to display the certificate details. For further details see the
description of the Mozilla web browser above.

**Web Application: Opera 7.54, OS: Windows 2000, Windows XP, Linux**

**Server Authentication:** The import of the certificates of PKI A is done with a PKCS #12 file
as in the previous scenarios. After editing the trust settings (warnings switched off), a
connection to a server which provides the full certificate-chain (including rootCA B and both
cross-certificates) is tried. Opera refuses to do a successful authentication. Instead a warning
is displayed:
It seems that Opera is stuck at the root certificate of PKI B and does not check for other trust paths. If the root certificate of PKI B is removed from the server-chain, it still does not work:

Obviously the cross certificate "old-with-new" (or in other words "B signs A") confuses Opera. After this cross certificate is removed, too, Opera establishes a secure connection to the server without any warnings. So Opera is not able to find the correct trust path for the full set of provided certificates. If the server only provides the old certificate chain of its PKI (leading up to the root certificate of PKI B), the cross certificate has to be imported into the browser. If only the cross certificate "new-with-old" is imported, the connection to the server is established without warnings. If additionally the cross certificate "old-with-new" is imported, Opera again produces warnings on connection establishment. After the expiry of PKI A, a secure connection does not work without warnings in any of the tried cases.

**Summary:** Opera is not good at finding the correct trust path that leads up to a trusted authority, if multiple paths lead different ways. When both cross certificates are imported into the browser, warnings are always displayed when trying to establish a secure connection. Authentication works only if the provided certificate-chain does not provide more than one possible path up to a trusted root certificate. So scenario 7 does not work for an Opera
Server Authentication: The first try with an imported root certificate of PKI A and a serverchain containing rootCA B, subCA B and both cross certificates does not work. After the import of the cross certificate "new-with-old" (in other words "A signs B") the chain can be validated correctly. This indicates some issues with following multiple trust paths. When the root certificate of PKI A is removed from the local certificate store, the chain cannot be validated again. This proves that chains are always validated up to the root certificate (otherwise the validation would already be successful when reaching the locally imported cross certificate).

When setting the date after the expiration of the old root certificate of PKI A (and the according cross certificate "new-with-old"), no trust in the new PKI B remains, so the cross certificates do not provide a permanent trust relation to the linked PKIs (see the following figure, note that the certificate chain is displayed as expired though the actual displayed certificate is still valid at that time, 27th of January, 2006).

As already tested in scenario 1, the manual import of single certificates works as expected, in larger setups CA certificates can be provided in the according KDE files (see konqueror tests
scenario 1).

**Summary:** Obviously Konqueror is not always able to validate certificate chains with multiple possible paths. Distributing the cross certificate to the clients provides a temporary solution to establish trust in the new root certificate. This trust only exists as long as the cross certificate is valid, after the expiry the root certificate of PKI B has to be configured manually as being trustworthy.
Results

In this chapter the results of all the single tests are presented in a short overview. For further details on single applications the according section from the previous chapter should be consulted. Overall the support of the tested scenarios was quite good. In the following paragraphs the results for the different scenarios are summarized. This summary is followed by a short descriptions of the observed application behaviour.

Results for the different scenarios

Scenario 1: The import of PKCS12 files is supported by every tested application. The feature of including some additional root-certificates in the PKCS12 file does not disturb most of the applications, it seems that only Konqueror refuses to import CA certificates from a PKCS #12 file (so even the CA certificate chain of the own PKI cannot be distributed together with the users certificate). Some applications require the user to acknowledge the import for all the included certificates, some ask the user to verify the fingerprint of each contained root certificate while other require manual editing of the trust-settings of each certificate after the import. So the only difficulty with this scenario is the initial trust settings for the imported CA certificates. It may simply be impossible for a user to find all the imported certificates in the list of trusted authorities and change their trust settings (especially if the imported certificates were not displayed during the import process). Unfortunately this scenario is only of use in a setting where PKCS #12 is already in use for distributing certificates of the own PKI.

Scenario 2: Obviously not every application can handle PKCS #7 files. The expected filename suffix for PKCS7 files differs from application to application, ".der" is the only accepted suffix for Mozilla, Thunderbird and FireFox, the Microsoft certificate manager uses ".p7b" or ".spc", for gpg/kmail ".p7m" is the correct naming, while Opera uses ".p7". The import of PKCS #7 files works flawlessly with the Microsoft certificate manager and Opera, while konqueror does not support the format at all. Though Mozilla, Thunderbird and Firefox are said to import PKCS #7 files, there is no file suffix applicable for it besides the usual ".der" for single certificate files, and the import stops after the first included certificate. So it seems that PKCS #7 is a good way of distributing additional root-certificates (even where no own PKI is in place), but it works only for environments using the Windows certificate manager or Opera.

Scenario 3: The method of linking different PKIs with a SuperCA supported by most of the tested applications. Only a few programs get confused by the replacement certificates for the former rootCAs or do not try to validate all the possible trust paths up to a trusted root certificate. Unfortunately the additional problem of distributing the SuperCA certificate and the replacement certificates has to be solved. A PKCS #7 could provide all the necessary certificates, but the issues of scenario 2 would have to be addressed as well.

Scenario 4: A cross-certificate can provide a link from the own rootCA to the rootCA of foreign PKIs. This seems to be a very elegant solution as in some cases it would not even be necessary to import the cross-certificate into the client applications (eg. if the cross-certificate
is included in the certificate chain of webservers or of signed emails). All the tested Microsoft applications and Mozilla, Thunderbird and Firefox can handle such cross certificates. The big advantage is that usually the trust-level of the cross certificate does not have to be edited manually, because for most applications it is sufficient if the superordinate root certificate is already trusted. This is usually the case because it is already necessary for the own PKI to work (remember: the issuer of the cross certificate is the rootCA of the own PKI). So this scenario is very promising when an own PKI is already in place. It could become a problem if the foreign PKIs issue the same kind of cross certificates to the own PKI: This would introduce cycles in the trust graph which seems to irritate some of the applications. See scenario 7 for further details on this.

**Scenario 5:** If not the whole foreign PKI is to be trusted but only a part of it, a cross certificate can be issued to a subCA of the foreign PKI. This way only certificates issued by this subCA are trusted. This works for most of the applications in the same way as scenario 4. There exists an additional issue with Opera and Konqueror related to the fact that the linking of both PKIs does not take place at the top level (root certificates) of the PKI. While validating a certificate chain provided by the communication partner it is not sufficient to follow up a single path as far as it gets and then trying to continue with the own, locally kept certificates. In this case multiple paths up to a root certificate exist and for each certificate on the way the application has to test if there is possible continuation within the certificates stored locally. Additionally there could be a problem with some of the applications if additional cross certificates from foreign PKIs to the own PKI exist (see scenario 7).

**Scenario 6:** All the issues of scenario 5 also apply for this scenario. Additionally Opera seems to have problems with the import of the cross-certificate which seems to interfere with the already imported subCA-certificate of PKI A. It does work with all the tested Microsoft applications and with Mozilla, Thunderbird and Firefox, though for the later three it could become a problem if additional cross-certificates from foreign PKIs to the own PKI exist (see scenario 7).

**Scenario 7:** This scenario was originally included in the tests because it could provide a means to roll out a succeeding PKI when the former PKI is about to expire. Surprisingly in these tests it showed the limited support of cross certificates when these introduce cycles in the trust graph. The scenario proposed in the PKIX standard [RFC 3280] makes use of two cross certificates, one in each direction between the root certificates of the old and the new PKI. So there exists a certificate from rootCA A issued to rootCA B and a second one issued by rootCA B to rootCA A, both building a very obvious cycle. Some applications were stuck immediately in hanging in this infinite loop while validating the certificate chain. So especially the cross certificate "old-with-new" (which enables users of the succeeding PKI B to validate signatures from the predecessing PKI A) seems to cause problems for users in PKI A. A solution could be to provide users of PKI A with the cross certificate "new-with-old" only while users of PKI B just get "old-with-new". For users of PKI A the trust in PKI B vanished immediately with the expiry of PKI A. In real life this would be the time to issue certificates from PKI B to them anyway, so it may not be a problem. A mechanism to ensure that the new root certificate of PKI B is installed locally as trusted root certificate while the cross certificate is still valid would be "nice to have". This way permanent trust in the
succeeding PKI B could be established. This behaviour was not implemented by any of the tested applications. An unexpected result was the checking of signatures of S/MIME emails. Most of the applications still held the signatures of users from PKI B valid, even after PKI A already expired. This seems to make sense because the signatures were made at a time when PKI A was still valid. On the other hand this may be a security issue because after years the keylength of PKI A could become critical enabling attackers to forge valid cross certificates of this PKI. Obviously Microsoft Outlook XP does not check the time of signature creation so all signatures created during the validity period of PKI B are held to be invalid after the expiry of PKI A. This was especially surprising because Microsoft Outlook Express did not show this behaviour. Which of these results was intended remains open.

Results for the different applications

Outlook Express 6.0 SP1: Outlook Express was the only email application supporting every scenario without restrictions. The import and handling of certificates is left to the Windows certificate manager, PKCS #12 and PKC #7 files are fully supported. This enables the simple distribution of all necessary certificates for the SuperCA scenario. Cross-certificates provide a means to link to different PKIs without any necessary actions on the client side, the cross-certificate can be provided in a signed email and is imported automatically into the Windows certificate store. In scenario 7 even after the expiry of PKI A the signatures created in the validity period of this PKI are still held to be valid.

Outlook XP: Outlook XP uses the Windows certificate manager for the handling of the certificates. Additionally it requires the user to import all recipients certificates in its own contact database. Only during this process are end user certificates imported into the certificate manager, so sending encrypted emails is only possible, if the recipients certificate was imported into the contact database. Everything else works flawlessly as with Outlook Express, the only difference being the signature validation in scenario 7. Outlook XP regards signatures from PKI B invalid after the cross-certificate is expired (though the signatures were created during the validity period of the cross-certificate).

Mozilla 1.7.2, Thunderbird 1.0 & Firefox 1.0: All three applications showed exactly the same behaviour in all tested scenarios and on all tested operating systems. The import of PKCS #12 works flawlessly, it requires the user to manually edit the trust settings for each imported certificate though. PKCS #7 seems to be only partially supported (and it seems that the developers of Mozilla object against different root-certificates in a single PKCS #7 file, see the email thread on this topic on https://bugzilla.mozilla.org/show_bug.cgi?id=236461). This behaviour makes distribution of different CA-certificates difficult which is the only challenge for scenario 3: Handling of a SuperCA works flawlessly. The same applies for scenario 4, 5 and 6, though a problem remains: When all involved rootCAs issue cross certificates to the other PKIs, loops are created in the trust graph. This is not handled correctly by all the Mozilla applications (see scenario 7). For certain applications cross certificate provide a means to link to different PKIs without any necessary actions on the client side (eg. when a webserver or a signed email provides not only the own PKIs certificate-chain, but also the cross certificate).
**Kmail:** Recent version of Kmail use GnuPG as backend for handling X.509 certificates. So the `gpg-agent` is queried to verify certificates and signatures. The complete S/MIME integration is impressive, OCSP and LDAP seems to be fully supported as well as configuration using a graphical user interface. The import of root certificates using PKCS #12 and PKCS #7 files works as expected, only the trust settings for root certificates have to be edited (which means editing a configuration file in this case). Alternatively the `gpg-agent` can be configured to ask the user if some new root certificates are trusted whenever they are used. The validation of certificate chains does not seem to follow all the possible paths up to a trusted root certificate, it looks as if GnuPG only chooses the first certificate found in the keyring that matches the issuer entry of the certificate to be checked. So depending on the order of the certificates in the local keyring, a signature may be verified or not. A bug report was sent to the developers of GnuPG, according to an email from Werner Koch this issue was solved in the most recent versions of gpgsm (Mail from 20.04.2005).

**Opera:** The observed behaviour of Opera was the same under all tested operating systems. It has a good support for the import of PKCS #7 files, PKCS #12 import worked flawless with the tested certificates. (In some different tests Opera refused to import some certificates which held a different set of extensions than specified by the HEPKI profile. This may as well be an issue of the PKCS #12 standard which seems to be very "flexible"). Nevertheless the import seems a little bit shaky (eg the list of trusted authorities includes the newly imported certificates only after restarting the Opera certificate manager), but the display of necessary information about the included certificates is very well. This impression seems to be supported by Opera not being able to import the cross certificates in scenario 6 and 7. Scenario 3, 4 and 5 do work mainly, Opera is very picky about the certificate chain provided by web servers. This can be a problem with server software that does not provide means to configure the transmitted certificate chain. In some cases it is necessary to provide only the server certificate (while all other certificates are imported into the browser), which in turn will not work with different client applications (which expect a normal certificate chain).

**Konqueror:** Konqueror does not support the import of multiple root-certificates from PKCS #12 files (only the included user-certificate and key were installed), PKCS #7 files are not supported at all. These restrictions complicate the linking of different PKIs, only single certificates can be imported. When the certificates are imported, scenario 3 and 4 work without restrictions. If cross certificates are used, not all possible paths to trusted root certificates are evaluated, so scenario 5 and 6 do not work as expected (depending on the certificates included in the transmitted certificate chain). The same holds true for scenario 7, which refused to work until a specially crafted certificate chain (without the certificates "rootCA B" and "old-signs-new", so only a single path to rootCA A is possible) was provided.

**Overview of results**

The following table summarizes all the presented results for the tested applications.
## Overview of Results

<table>
<thead>
<tr>
<th>Application</th>
<th>Operating System</th>
<th>Scenario 01</th>
<th>Scenario 02</th>
<th>Scenario 03</th>
<th>Scenario 04</th>
<th>Scenario 05</th>
<th>Scenario 06</th>
<th>Scenario 07</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outlook Express 8.0 SP</td>
<td>Windows 2000</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Outlook XP</td>
<td>Windows XP</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Mozilla 1.7.2</td>
<td>Windows 2000</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y N Y N</td>
</tr>
<tr>
<td>Thunderbird 1.0</td>
<td>Windows 2000</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y N N N</td>
</tr>
<tr>
<td>Kmail 1.7 (KDE 3.3)</td>
<td>SuSE Linux 9.2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N N N N</td>
</tr>
<tr>
<td>Internet Explorer 6.0 SP</td>
<td>Windows 2000</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y Y Y Y</td>
</tr>
<tr>
<td>Mozilla 1.7.2</td>
<td>Windows XP</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y N N Y</td>
</tr>
<tr>
<td>Firefox 1.0</td>
<td>Windows 2000</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y N N N</td>
</tr>
<tr>
<td>SuSE Linux 9.2</td>
<td>Windows XP</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y N N Y</td>
</tr>
<tr>
<td>Opera 7.54</td>
<td>Windows 2000</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y N Y N</td>
</tr>
<tr>
<td>Konqueror (KDE 3.3)</td>
<td>SuSE Linux 9.2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N N N N</td>
</tr>
</tbody>
</table>

1: After expiry of old PKI, encryption is not possible
2: Change of defaults for certificate-store necessary
3: After expiry of old PKI, signature verification fails
4: After import, manual editing of root-trust necessary
5: Only 1st certificate imported
6: Manual import of multiple certs
7: Import of cross-cert "old-with-new" fails
8: Application hangs if "old-with-new" imported
9: Configuration of gpg-agent necessary
10: Manual editing of root-trust necessary
11: Not all chains evaluated, false negatives
12: Depending on results of verification (see 11)
13: After expiry of old PKI, warnings are displayed
14: After import, switch off "display warnings"
15: Adapted serverchain necessary
16: Import of cross-cert works only with tricks
17: Only usecert and key are imported
18: rootB & "old-with-new" in chain: false negatives

84
Summary and future prospects

Some of the possible scenarios to link different PKIs have been tested with the most common applications. Scenarios 1 and 2 examined the ways to distribute the necessary certificates as files that can be easily imported into the client applications. The support for PKCS #12 is very common because it provides the only means to import the users secret key and certificate when key generation is done by the certification authority. In most cases it is possible to distribute additional root certificates though these are usually not trusted after the import. So in some cases manual editing of the trust of all certificates is necessary. Distribution of PKCS #7 files which hold the certificates of different rootCAs suffers from the missing support in some of the most common applications (eg Mozilla, Thunderbird, Firefox), while the support in Microsoft software and Opera is good. The linking of different PKIs using a SuperCA, which in turns issues replacement certificates for all the former root certificates could be a good solution when multiple PKIs have to be linked (eg. for a bridge CA). Unfortunately it depends on the distribution of the SuperCA certificate and all the replacement certificates, so all the restrictions of scenario 1 and 2 usually apply.

Cross certificates offer the most transparent linking method, in some cases manual import of certificates may not be necessary at all for the client applications (eg when the cross certificate is transmitted in the certificate chain of a webserver or a signed email). Cross certificates rely on the trust into the own PKI (which is usually in place anyway) so editing trust settings for certificates is not required. Additionally the control of the trust in foreign PKIs still remains in the own PKI (because the cross-certificate belongs to the own PKI and can always be revoked, see [HaPe2002]).

Cross certificates offer a more detailed granularity of trust into foreign PKIs. It is possible to issue cross certificates for parts of the foreign PKIs only (scenario 5) and even to link the foreign PKI to only a subCA of the own PKI (scenario 6). Unfortunately these certificates cannot be handled by some of the common applications. The certificate validation algorithms seem to ignore the fact, that multiple paths can lead up to a trusted root certificate. Microsoft applications seem to have the most advanced support of cross certificates, even additional extensions can be used to control the trust in foreign PKIs (using PolicyMappings, untested for this report).

The special scenario of using cross certificates to provide a link from an expiring own PKI to its successor still causes problems in some of the applications because the two cross certificates create a cycle in the trust graph. Obviously the validation algorithms in the applications do rely on the graph to be only a tree and do not check for loops.

The simple cross certificates from scenario 4 could prove to be a good solution for linking PKIs already. It is expected that the problem of GnuPG will be eliminated soon, so the most common applications do already support this cross-certificates (hopefully the support for multiple links and even loops in the certificate chains introduced by cross-certificates will grow in time, right now only Microsoft seems to have the complete functionality deployed already).

So especially in the Microsoft context some further tests with PolicyMappings would be promising. Most of the other applications could still need a little bit of homework to be done.
Appendix A: Certificate Profiles

The following tables show the X.509 extensions specified by the HE PKI-Lite profiles ([HEPKI 2004a], [HEPKI 2004b]) and their usage in the test certificates. For each given field, the tables state if it is specified by the related HE PKI-Lite profile ("Specified") and if it is used according to the specification ("Used").

### HE-PKI Lite CA Certificate Profile: X.509v3

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
<th>Specified</th>
<th>Used</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>0x2</td>
<td>Y</td>
<td>Y</td>
<td>Version 3 certificates</td>
</tr>
<tr>
<td>Serial Number</td>
<td>unique integer</td>
<td>Y</td>
<td>Y</td>
<td>unique integer for each CA</td>
</tr>
<tr>
<td>Signature Algor.</td>
<td>SHA1/RSA</td>
<td>N</td>
<td>Y</td>
<td>SHA1/RSA recommended</td>
</tr>
<tr>
<td>Issuer</td>
<td>DN</td>
<td>Y</td>
<td>Y</td>
<td>simple entries, no DC naming</td>
</tr>
<tr>
<td>Validity Time</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>validity periods of signed certificates fully included in validity periods of signing certificate</td>
</tr>
<tr>
<td>Subject DN</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>simple entries, no DC naming, Email Address</td>
</tr>
<tr>
<td>Public Key</td>
<td>N</td>
<td>Y</td>
<td></td>
<td>Minimum 2048 bit</td>
</tr>
</tbody>
</table>

### HE-PKI Lite CA Certificate Profile: Certificate Extensions

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
<th>Specified</th>
<th>Used</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyUsage</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basic Constraints</td>
<td>CA=true</td>
<td>Y</td>
<td>Y</td>
<td>No PathLength, Critical</td>
</tr>
<tr>
<td>CRL Distribution Points</td>
<td>N</td>
<td>N</td>
<td></td>
<td>Not subject of these tests</td>
</tr>
<tr>
<td>Certificate Policy</td>
<td>N</td>
<td>N</td>
<td></td>
<td>Could point to HE PKI-Lite Policy, not subject of these tests</td>
</tr>
<tr>
<td>Subject Alternative Name</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>Email Address</td>
</tr>
<tr>
<td>CPS Pointer</td>
<td>URI of CPS</td>
<td></td>
<td></td>
<td>Not subject of these tests</td>
</tr>
</tbody>
</table>

2 The HE PKI-Lite CA profile recommends validity periods that are significantly longer than those of the issued end entity certificates. Because the generated CA certificates are for testing only, this is not necessary. Still the validity periods of the issued end entity certificates are fully included in those of the signing CA certificate.
### HE-PKI Lite CA Certificate Profile: Certificate Extensions

| Other fields | ? | CAs may include other elements as needed, not subject of these tests |

### HE-PKI Lite End Entity Certificate Profile: X.509v3

<table>
<thead>
<tr>
<th>FieldName</th>
<th>Value</th>
<th>Specified</th>
<th>Used</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>0x2</td>
<td>Y</td>
<td>Y</td>
<td>Version 3 certificates</td>
</tr>
<tr>
<td>Serial Number</td>
<td>unique integer</td>
<td>Y</td>
<td>Y</td>
<td>unique integer for each CA</td>
</tr>
<tr>
<td>Signature Algor.</td>
<td>SHA1/RSA</td>
<td>N</td>
<td>Y</td>
<td>SHA1/RSA recommended</td>
</tr>
<tr>
<td>Issuer</td>
<td>DN</td>
<td>Y</td>
<td>Y</td>
<td>simple entries, no DC naming</td>
</tr>
<tr>
<td>Validity</td>
<td>Time</td>
<td>Y</td>
<td>Y</td>
<td>validity periods fully included in validity periods of signing CA certificate</td>
</tr>
<tr>
<td>Subject</td>
<td>DN</td>
<td>Y</td>
<td>Y</td>
<td>simple entries, no DC naming, Email Address</td>
</tr>
<tr>
<td>Public Key</td>
<td>N</td>
<td>Y</td>
<td></td>
<td>Minimum 1024 bits, 2048 bits are used</td>
</tr>
</tbody>
</table>

### HE-PKI Lite End Entity Certificate Profile: Certificate Extensions

<table>
<thead>
<tr>
<th>FieldName</th>
<th>Value</th>
<th>Specified</th>
<th>Used</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyUsage</td>
<td>N</td>
<td>Y</td>
<td></td>
<td>DigitalSignature, KeyEncipherment</td>
</tr>
<tr>
<td>Basic Constraints</td>
<td>CA=false</td>
<td>Y</td>
<td>Y</td>
<td>Critical</td>
</tr>
<tr>
<td>CRL Distribution</td>
<td>N</td>
<td>N</td>
<td></td>
<td>Not subject of these tests</td>
</tr>
<tr>
<td>Points</td>
<td>Certificate Policy</td>
<td>N</td>
<td>N</td>
<td>Could point to HE PKI-Lite Policy, not subject of these tests</td>
</tr>
<tr>
<td>Subject</td>
<td>Email</td>
<td>Y</td>
<td>Y</td>
<td>Email Address</td>
</tr>
<tr>
<td>Alternative Name</td>
<td>Other Name</td>
<td>N</td>
<td>N</td>
<td>Microsoft applications use this for authentication, if not present, the CN is used (which then must be unique). This is the case for the test certificates.</td>
</tr>
<tr>
<td>CPS Pointer</td>
<td>URI of CPS</td>
<td></td>
<td></td>
<td>Not subject of these tests</td>
</tr>
<tr>
<td>Other fields</td>
<td>?</td>
<td></td>
<td></td>
<td>CAs may include other elements as needed, not necessary for these tests</td>
</tr>
</tbody>
</table>

The fields "Policy Mappings" and "Policy Constraints" were not used in the tests.
Appendix B: Test Framework Scripts

For the creation of the test certificates some Perl scripts were written. These are able to build complete test PKIs with a few commands, so the process of building certificates for every single test is highly automated. This way an identical test environment is guaranteed for further tests, that may be conducted by other persons. The following section introduces the modules and gives short explanation about their functions. The second section describes the necessary steps to create the test certificates.

Introduction to the modules and scripts

The test scripts are written in Perl. The full set consists of one script per test scenario, which builds all the necessary PKIs and files for the scenario. These scripts in turn use some Perl modules which provide the necessary complex functions of X.509 handling. Table 5 shows the according Perl modules:

<table>
<thead>
<tr>
<th>Script</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>HELPERS.pm</td>
<td>Some utility functions for creating names for tempfiles etc.</td>
</tr>
<tr>
<td>OpenSSL.pm</td>
<td>A module for creation and management of X.509 keys and certificates using OpenSSL.</td>
</tr>
<tr>
<td>config.pm</td>
<td>Loading and saving of configuration files for all scripts.</td>
</tr>
<tr>
<td>x509_export.pm</td>
<td>Exporting X.509 keys and certificates into files (PKC12, PKCS7)</td>
</tr>
<tr>
<td>x509_hierarchy.pm</td>
<td>Creation of CA hierarchies including root CAs, subordinate CAs and user certificates.</td>
</tr>
<tr>
<td>x509_mail.pm</td>
<td>Creation of signed S/MIME emails, output as S/MIME mail (which can be sent using an MTA like mailx) or as MBOX file</td>
</tr>
</tbody>
</table>

Table 3: Support Modules for Test Scripts

Using x509_hierarchy.pm a complete CA can be constructed using only a few commands. Table 4 illustrates the usage of the module. The given code creates a root CA (ca_type set to ca_ca) in the specified directory (/path/to/cadir/testroot). Creation of a related subCA would only require to additionally specify the path of the root CA (and eventually the password of the CA, if not the default password for all the test CAs). For root CAs the initial serial number should always be set to at least 01, otherwise the selfsigned root certificate as well as the first issued CA certificate will get the serial number 00, so the serial numbers of this CA would not be unique. Scripts using these functions to create PKIs have to take care for the correct validity periods (the period of a subCA should always be fully included in that of the related root CA), this is usually achieved by decreasing the number of days by one.
```perl
#!/usr/bin/perl
use x509_hierarchy;
$dn = {   C     => 'DE',            ST    => '.
          L     => '.',             O     => 'Test Organization A',
          OU    => '.',             CN    => 'Test Root CA A1',
          EMAIL => 'test@test.org'
};
x509_hierarchy::create_ca(
    "/path/to/cadir/testroot",
    
    
    { days           => 730,
      ca_type        => 'ca_ca',
      dn             => $dn,
      subjectkeyid   => 'hash',
      authoritykeyid => 'keyid',
      serial         => '01'
    }
);```

Table 4: Usage of x509_hierarchy.pm module

Table 5 shows the usage of the module x509_export.pm, a PKCS #12 file is created containing a user certificate, the users key and the certificate of the subCA. For CA certificates it is sufficient to specify only the path to the CA and the type CACERT, because the CA certificate has a fixed name in the CA folder. For user keys and user certificates, the distinguished name of the subject has to be specified to find the correct certificate. Userkeys can only be exported to PKCS #12 files. They require an argument pass containing the password of the secret key, so the key can be decrypted and included in the file. The functions gen_p7 and gen_p12 handle a variable number of arguments, so it is possible to include additional certificates in the generated files.

The usage of the module x509_mail.pm is shown in table 6. A signed mail is created using the textfile email.txt as mail body, the output is written to file email.signed. The mail is signed using the certificate of user B, the full certificate chain is included. The file is created as an MBOX file, so a "From" line containing the senders name and the actual date is prepended.
#!/usr/bin/perl
use x509_export;

# [ left out: set the subject name of the user certificate and key, variable $dn_user ]
my $userkey = {
    path => "/path/to/cadir/testsub",
    type => 'KEY',
    dn   => $dn_user,
    pass => 'test'
};
my $usercert = {
    path => "/path/to/cadir/testsub",
    type => 'CERT',
    dn   => $dn_user
};
my $rootca = {
    path => "/path/to/cadir/testroot",
    type => 'CACERT'
};
my $subca = {
    path => "/path/to/cadir/testsub",
    type => 'CACERT'
};
x509_export::gen_p12("/path/to/export/test-01.p12",
    $rootca, $subca, $usercert, $userkey);

Table 5: Usage of x509_export.pm module

#!/usr/bin/perl
use x509_mail;

# [ left out: set the subject name of the user certificate and key, variable $dn_user ]
my $userkey = {
    path => "/path/to/cadir/testsub",
    type => 'KEY',
    dn   => $dn_user,
    pass => 'test'
};
my $usercert = {
    path => "/path/to/cadir/testsub",
    type => 'CERT',
    dn   => $dn_user
};
my $rootca = {
    path => "/path/to/cadir/testroot",
    type => 'CACERT'
};
my $subca = {
    path => "/path/to/cadir/testsub",
    type => 'CACERT'
};
x509_mail::create_signedmail(
    {  subject   => "PKI-Linking Testmail, Scenario 01",
        from => "sender@test.org", to => "receiver@test.org",
        infile  => "/path/to/email.txt",
        outfile => "/path/to/email.signed",
        from_cert => $usercert, from_key => $userkey,
        mbox => 1 }, $subca, $rootca);

Table 6: Usage of x509_mail.pm module
For each of the linking methods to be tested, a script has been setup that produces all the necessary files for the test. The complete PKIs are created and the necessary keys and certificates are exported to files that can be imported into client applications (file formats are PKCS #12 and PKCS #7 depending on whether secret keys are necessary or not). Additionally, signed emails are created to simplify the necessary test environment. A central configuration file holds the values to be used for the subject entries of the according certificates.

The scripts require the environment variable `PKI_LINK_DIR` to be set to the directory containing the scripts. They read the configuration file `linking.conf` in this directory and use the modules contained in the subdirectory `lib`. The scripts are described in detail in the sections of the conducted tests.

---

**Setting up the test PKIs**

First the configuration file `linking.conf` has to be edited according to the local environment. The variable `PKIdir` is used to specify, where the directories of the PKIs for all the tests are created. If testing actually requires the sending and receiving of emails, it may be necessary to edit the email addresses in the user certificates (variables `dn_user_a` and `dn_user_b`) according to the used email addresses of the sender and receiver. The same has to be done for the CN entries for webserver certificates (variables `dn_server_a` and `dn_server_b`).

Before running the scripts, the environment variable `PKI_LINK_DIR` has to be set to the directory containing the scripts. The following commands can be used to create a full set of test PKIs and import files for each scenario. (The commands assume using `BASH`, a `CSH` would require the use of `setenv` instead of `export`.)

```
export PKI_LINK_DIR=/home/myname/pki_linking/software
cd $PKI_LINK_DIR
edit linking.conf
./build_scenario_00.pl
./build_scenario_01.pl
./build_scenario_02.pl
./build_scenario_03.pl
./build_scenario_04.pl
./build_scenario_05.pl
./build_scenario_06.pl
./build_scenario_07.pl
```
After this all the PKIs and files can be found in the `pkidir` directory specified in the configuration file. Each scenario is created in its own subdirectory.

Bibliography


CrCo2004: Creative Commons, Attribution-NonCommercial-ShareAlike License, Version 2.0, 2004. [http://creativecommons.org/licenses/by-nc-sa/2.0/](http://creativecommons.org/licenses/by-nc-sa/2.0/)


