Setting up the Apache web server to support PHP based strong authentication

Tomi Kontio

Report B/2007/1

UNIVERSITY OF KUOPIO
Department of Computer Science
P.O.Box 1627, FI-70211 Kuopio, FINLAND

ABSTRACT

This report describes the steps necessary in order to enable certificate based authentication using the Apache web server. As a client certificate we use the certificate issued by the Finnish Population Register Centre (PRC). The report includes instructions how to create a certificate authority and a server certificate for secure SSL connection.

1. INTRODUCTION

This report is a part of Tomi Kontio's Master's Thesis. The subject of the Master's Thesis is "Strong authentication of users in a PHP-based web system". The supervisor of the Thesis and this report is Licentiate of Philosophy Marko Hassinen from the University of Kuopio. Most of the work concerning the Thesis was made during summer 2006, and the process was finished in the beginning of 2007. The thesis and its writing process were funded by the department of computer science of the University of Kuopio.

The report describes the efforts made during the experimental part of the Master's Thesis. The goal of the research was to enable strong authentication of users. The report forms a technical addendum to the Master's Thesis.

This report follows the process of creating a certificate based authentication mostly in chronological order. The focus of this paper is in providing the solutions to the technical difficulties encountered during the research process thus helping other people with setting up similar systems.

The section 2 "Research software" introduces the most important software needed during the research process. The section 3 "Certificates" describes the procedure of obtaining certificates from the Population Register Centre and provides a sample from the content of a certificate and the usage of OpenSSL. The section 4 "Creating own certificate authority and server certificate" follows a process of creating a new certificate authority and server certificate using the OpenSSL. The section 5 "Configuration files of the Apache web server" provides the contents and modifications made to the default configuration files of the Apache web server. The section 6 "PHP

scripts” explains the technically most relevant scripts and their content. The reasoning behind certain technical decisions is explained in section 6.

2. RESEARCH SOFTWARE

As a first decision we did prior to research was to use Apache as the web server in our research. Apache has been the most popular web server in the world since 1996 [1]. Apache was a natural choice for the research system. We chose the Apache version 2.2.3, which was the newest version available at the time of research (11/2006). A more difficult decision to make was whether to use a pre-compiled package or to compile the application from the source code.

The purpose of the research in the Master's thesis was to find out how one can set up a web service where the authentication is done using user certificates. Considering this background it seemed wiser to choose the pre-compiled installation package rather than try to compile the package ourselves. The chosen installation package from the Apache Lounge web site [2] also included quite many of the required modules for Apache: *mod_ssl* module for encrypted SSL-connections and PHP interpreter with enabled LDAP-connection support. The most important reason to choose a ready-to-install package was to save time to be able to finish the thesis according to the schedule.

The Apache Lounge is the home site of the Apache Windows 32 port. The site offers the main Apache application and numerous modules for Apaches running on 32 bit Windows operating systems. The Apache Lounge site has an excellent installation tutorial [3]. The tutorial includes all the necessary steps to install and set up Apache and PHP on a Windows environment.

After the installation was complete we came to a conclusion that the tutorial is exhaustive. The guide helped us to avoid all the common mistakes and misconfigurations during installation thus leaving us more time to concentrate on the main goal. Especially the snippets from the correct PHP-configuration were valuable because even the installation guide from the PHP home site [4] has outdated information about the correct installation procedure and directives to be used. Listing 1 shows the correct way to enable PHP support in the Apache configuration files.

```
LoadModule php5_module "c:/php5/php5apache2.dll"
AddHandler application/x-httpd-php .php
# configure the path to php.ini
PHPIniDir "c:/php5"
```

Listing 1. Correct Apache configuration file directives to enable PHP scripting support

3. CERTIFICATES

The first task involving certificates was to store the certificates of the Finnish Population Register Centre (PRC). Both the Population Register Centre Root CA Certificate and the CA for Citizen Qualified Certificate are available online at the web pages of the PRC [5]. The certificates were saved using the Mozilla Firefox web browser. The intent was to save these two certificates and install them into the Apache web server. If the certificates are installed available to Apache it could read those certificates. Then Apache could also decipher the certificate from the client machine. Unfortunately it turned out not to be that simple to configure Apache to read these certificates.

Initially, when the certificates were saved from the PRC web site they were in the DER (Data Encoding Rules) encoding [10]. The Apache web server is only able to utilize PEM (Privacy Enhanced Mail) encoded certificates. The first task was to encode certificates in the correct form. The easiest way to switch encoding was to use the certificate export tool provided by Windows XP. The certificate export tool can be started by double-clicking the downloaded *.crt*-file and selecting the Information tab.

It is possible to print the contents of the certificates using the OpenSSL tools. The OpenSSL and its documentation are available from the OpenSSL project [6]. The OpenSSL command to print the PRC Citizen Qualified Certificate root certificate in a human readable format is

```
openssl x509 -in vrkcqc.crt -text
```

Parameters explained:

x509	using the x509 tools
-in	specifies what certificate we want to process
-text	tells the OpenSSL to print out the contents of the certificate in plain text.

PRC Root certificate can be printed in a similar way using the command:

```
openssl x509 -in vrkrootc.crt –text
```

The output of the command using the PRC Citizen Qualified Certificate can be read in Appendix A. The PEM encoded part of certificate is the one that Apache understands

and is able to use for authentication. The Apache configuration file directives that are used to enable two-way authentication are described in section 5.

4. CREATING OWN CERTIFICATE AUTHORITY AND SERVER CERTIFICATE

The PRC sells server certificates. The main purpose of the server certificates is to authenticate the server to the client and to enable a secure SSL-connection between the server and the client. Considering the economical resources of the Master's Thesis it was not possible to acquire a server certificate from the PRC. Therefore we had to create our own server certificate. To make the certificate acquirement procedure more realistic we also created our own certificate authority (CA). Now we did not have to self sign the server certificate and the whole process became more like the real life situation. We could generate the certificate signing request (CSR) for the server certificate and sign the CSR with our CA instead.

The CA certificate was created using the following command:

```
openssl req -new -x509 -out ca.crt
```

Parameters explained:

req	using the certificate signing request commands
-new	new signing request
-x509	instead of creating an actual CSR we want to create a self signed certificate
-out	the name of the certificate we are creating

The contents of our self signed certificate printed using the OpenSSL tools can be found in Appendix B. Once we had created our CA certificate, we could use it to sign other certificates.

The next step was to create a certificate signing request for our server certificate. The following command creates a CSR ready to be signed:

```
openssl req -new -out server.csr
```

Parameters explained:

req	using the certificate signing request commands
-new	new certificate signing request
-out	name of the CSR to create

The only difference between this command and the command we used to create the CA certificate is that this time we did not self sign the certificate. We generated the CSR we

could send to the CA for signing. The certificate signing request is PEM-encoded. For example the one we created looked like this:

```
-----BEGIN CERTIFICATE REQUEST-----
MIIB8DCCAVkCAQAwga8xCzAJBgNVBAYTAkZJMQ8wDQYDVQQIEwZLdW9waW8xDzAN
BgNVBAcTBkt1b3BpbzEdMBsGA1UEChMUVW5pdmVyc2l0eSBvZiBLdW9waW8xJzAl
BgNVBAsTHkRlcGFydG1lbnQgb2YgY29tcHV0ZXIgc2NpZW5jZTESMBAGA1UEAxMJ
bG9jYWxob3N0MSIwIAYJKoZIhvcNAQkBFhNrb250aW9AaHl0dGkudWt1LmZpMIGf
MA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC+ib10E6idE2h7ZQGdgQE+xdFNivLr
BH6qoIakta9FONvPzbFwy7uIoTD9nnrRbEL8jHOxIlmZJP8cfed88FR2ZCHyc8i6
4tjo1VpczZQrwmAl2l9ZhyAVGvqXBUYzr06JvHzMU913+12hAFkQKHi2nI0VSlPI
zqNmYLzB8belLQIDAQABoAAwDQYJKoZIhvcNAQEFBQADgYEAnyM0x2G4wJOEn1G5
NCaj/27wP1kOMGrz37ZYTQSFqZFzQ+WNXv+WFXXacgHV0WCEbCrkFRvqXWziPCdz
ZfyhHoO6GqdDgno/z81ykugFPUmBBcYgAX4/CIlmV6EzAPvEflbIstV2hR7CRbG4
tBdUuWN7TIg5zHSdZ88gY0OCu2U=
-----END CERTIFICATE REQUEST-----
```

Listing 2. Certificate signing request

Now we could sign the CSR using the command

```
openssl x509 -CA ca.crt -CAkey privkey.pem -in server.csr -
req -set_serial 1 -out server.crt -days 3650
```

New parameters explained:

-CA	Certificate authority certificate we want to sign with
-CAkey	Private key file of the CA
-req	Instead of X509-certificate the input will be a certificate signing request
-set_serial	serial number of the certificate
-days	Period of validity of the certificate

There was still one more step to do before the Apache web server would accept the server certificate and its private key. Since the previously created private key is RSA encrypted by default, we had to decrypt it. If we were using a Unix based operating system Apache would prompt for the key of the private key upon every start up. The Windows version does not have this functionality. The Apache log files are the only place to find out this fact. If we tried to start the Apache web server using the encrypted private key the following log entry was generated:

```
[Fri Nov 17 15:02:58 2006] [error] Init: SSLPassPhraseDialog builtin
is not supported on Win32 (key file
C:/apache2/conf/ssl.crt/server.key)
```

The decryption of the key was achieved by using the command:

```
openssl rsa -in server.key -out server_dec.key
```

After the key is decrypted one has to be very cautious with it. Everyone can read the content of such a secret key because it really is in plaintext.

We have seen all the parameters before so the explanation is not necessary. Now we had completed all the necessary steps to start building our own system. All we needed to do was to copy the server certificate and the unencrypted key to a place where Apache can find them. We used the recommended default directory *Apache/conf/ssl.crt*.

5. CONFIGURATION FILES OF THE APACHE WEB SERVER

There were a couple of things that we needed to set up in order to enable SSL connection and strong client authentication. We needed to modify the Apache configuration files. In our Apache distribution the general settings were in the *httpd.conf* file and all SSL related directives were in a separate *httpd-ssl.conf* file.

Only few modifications were necessary to the general configuration file. Listing 3 shows the most significant modifications made to the *httpd.conf* file. The SSLRequireSSL directive in the Directory container forces the machines to use SSL connection always when accessing the specified directory. The directory was the script execution directory *c:/Apache2/cgi-bin* in our case. Listing 4 indicates the include directive which needed to be uncommented in order to activate the separate SSL configuration file.

```
<Directory "c:/Apache2/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
# Scripts are only allowed with SSL connection
    SSLRequireSSL
</Directory>
```

Listing 3. Apache directives for script directory

```
# Secure (SSL/TLS) connections
# Uncommented
Include conf/extra/httpd-ssl.conf
```

Listing 4. Separate configuration file had to be enabled by uncommenting it

The activation of the certificate authority certificates required much more modifications to Apache configuration file *httpd-ssl.conf* than to the *httpd.conf* file. As the modifications were numerous, the whole configuration file is included in Appendix C. The comments are left as they were in the actual configuration file to make the demonstration more understandable.

Most of the required directives were present in the *httpd.conf* file by default but they were commented out. In addition to uncommenting them they also were modified so that they point to the correct location of each file.

One thing that is really worth mentioning from the Apache configuration file is the Location container. It was possible to specify the directives in the Location container so that the server became totally inaccessible. By specifying *SSLVerifyClient require* and *SSLVerifyDepth 2* outside the Location container the server requested the client to authenticate itself at the start of each new connection. This all happened before the client and server had finished the handshake protocol of the SSL. The situation became a deadlock: the client refused to send its certificate to the server because there was no secure connection and the server refused to open up an SSL connection because the client did not send its certificate. The worst part in this situation was that the client's web browser presented only a pop up error message which consisted of a single word: "-12227". Nothing else happened.

The message did not help the troubleshooting at all and it was really hard to tell what part of the configuration went wrong. The problem was solved partially by accident when we saw a related article on the Mozilla developer's forum [7]. The error code "-12227" turned out to mean "*SSL_ERROR_HANDSHAKE_FAILURE_ALERT: SSL peer was unable to negotiate an acceptable set of security parameters*". At that point we were amazed why that piece of information couldn't appear to the message dialog in the first place. Sure it isn't good practice to give highly technical error explanations to the end user but certainly giving out pure error codes isn't either.

The *php.ini* file defines all the options of the PHP script interpreter. Only very few modifications were necessary in order to enable the scripting support and strong authentication. The most important operation was to enable required extension libraries for LDAP-support and SSL-related tools. Appendix D has the contents of the *Windows Extensions* part of the *php.ini* -file. The bolded lines define the extension libraries which needed to be enabled, *php_ldap.dll* and *php_ssl.dll*.

6. PHP SCRIPTS

From a technical point of view the most interesting PHP scripts created during this research are *cert_info.php* and *crl.php*. The *cert_info.php* prints the contents of the client certificate, the certificate authority certificate and the server certificate from the environment variable array called $_SERVER. The *crl.php* script fetches the updated certificate revocation list every half an hour from the PRC LDAP server.

The *cert_info.php* answers the question about how the fields of the certificates are accessed from a PHP-script. All of the fields are automatically put into a global variable array called $_SERVER. The array is global so it can be accessed everywhere. The output in Appendix E is achieved by looping through the $_SERVER array with the PHP function *foreach()* and printing the key-value pairs into a table. Note that only array entries related to certificates are included in the appendix.

The *crl.php* is included in Appendix F. The script fetches the new certificate revocation list from the PRC LDAP server. After the CRL is downloaded it is converted from DER encoding to PEM encoding because Apache is able to utilize only PEM encoded certificates and certificate revocation lists. When the new CRL is downloaded and put in the Apache configuration directory, the only thing left to do is to restart Apache. Every operation we have done so far in this script to download and to install the CRL can be done by running the script as an Apache module but we cannot restart the Apache itself. That was the main reason why this operation was made in a separate script.

The most difficult part of making this script to work was the LDAP connection and handling of the downloaded CRL list. The connection itself was easy to create utilizing the documentation offered by PRC [8]. The difficulties began when we tried to create a solution which included parsing the downloaded CRL. It took a lot of time but did not give us any results. It turned out to be totally impossible to parse the newly downloaded list directly in the CGI PHP script.

The idea behind the functionality of downloading the CRL list from the CGI PHP script was that the CRL list will be always up-to-date. When the CRL is downloaded upon every page request it cannot contain expired information. However, the fact the PRC updates its list once in 30 minutes made this reasoning wrong. Using this idea would

also lower the usability of the system because downloading the list is a time consuming task.

The other fact that caused a lot of trouble was that the CRL list needs to be read in binary mode from the LDAP result set. The PHP function *ldap_get_values_len*() has to be used instead of *ldap_get_values*(). Unless the binary mode is used the result set will be invalid. It seems that the CRL contains a bit sequence which is interpreted as end-of-input marker if binary mode is not used. In the PRC documentation there were no clues about this problem.

Since the Windows operating system doesn't include a command line automation tool like Unix Cron we needed to find one. We decided to use freeware software nnCron LITE, which offers identical functionality as the aforementioned Cron. The nnCron LITE is developed by Nicholas Nemtsev and is free to use for noncommercial purposes [9]. The other possibility would have been the use of the Windows task scheduler but we did not want to limit the usability of the research system only to Windows operating systems.

The scheduler application nnCron LITE was very easy to use. All we needed to use was to add one line to the scheduler's configuration file *cron.tab*. The line consists of the name of the application to run at set intervals and a specification of the interval. The *cron.tab* file in our system is presented in Listing 5.

```
# CRONTAB FILE
# Classic crontab format:
# Minutes Hours Days Months WeekDays Com

*/30 * * * * c:\apache2\crl\update.bat
```

Listing 5. Schedule file *cron.tab*

The file included a guide of how to use cron-files. The second line contains information about when to launch the scheduled application and what application to launch. The contents of the batch script *update.bat* are shown in Listing 6.

```
c:\php5\php c:\apache2\crl\crl.php
httpd -k restart
```

Listing 6. Batch file *update.bat*

The batch file is extremely simple. First it launches the *crl.php* described in Appendix F and after that restarts the Apache web server. The Apache web server is restarted gracefully meaning that threads serving client machines won't be interrupted during the restart. By running this batch file periodically the CRL list stays up-to-date.

The *crl.php* script uses external system command to achieve its goal. At first we though we could cope with the functions included in the OpenSSL tool package of PHP. After multiple tries that turned out to not be a feasible solution. The encoding of the CRL list was much simpler to do using the OpenSSL tools directly than using the OpenSSL functions of the PHP. The PHP function *exec*() was suitable for using the external OpenSSL tool commands from the PHP script.

In order to use *crl.php* in your own system you need to modify the constant variables which refer to the directory path in the file system. That was the reason the file path and parameter list were put into variables instead of putting them directly into function calls. It is much easier to replace the contents of a variable than to find and replace the parameter list of a function call.

As you may have noticed the system consists of many small pieces, but there is one thing common to all of them: they are simple and straightforward. There are no complex decisions to make or technical difficulties to overcome. Yet the basic system described here is ready to be used in a real life system.

References

[1] Netcraft: *Web server survey archives*. 1996-2006. Referenced in 26.12.2006. Available from: http://news.netcraft.com/archives/web_server_survey.html

[2] Apache 2 on Windows Support & Consulting: *Apache Lounge*. 2006. Referenced in 26.12.2006. Available from: http://www.apachelounge.com/

[3] *Apache 2 :: Apache and PHP – a fast, reliable and proven setup*. 2006. Referenced in 26.12.2006. Available from: http://www.apachelounge.com/forum/viewtopic.php?t=570

[4] PHP: Hypertext Preprocessor. 2001-2006. Referenced in 26.12.2006. Available from: http://www.php.net

[5] Population Register Centre: *fineid.fi site -technical information about electronic identity*. 2006. Referenced in 26.12.2006. Available from: http://www.fineid.fi

[6] OpenSSL Project: *the Open Source Toolkit for SSL/TLS*. 2006. Referenced in 26.12.2006. Available from: http://www.openssl.org/

[7] Cotter S.: *SSL Reference*. 2000. Referenced in 26.12.2006. Available from http://www.mozilla.org/projects/security/pki/nss/ref/ssl/sslerr.html

[8] Population Register Centre: *FINEID - S5 directory specification v2.1*. 2004. Referenced in 26.12.2006. Available from: http://www.fineid.fi

[9] Nemtsev N.: *nnSoft: nnCron*. 2002. Referenced in 26.12.2006. Available from: http://www.nncron.ru/

[10] Population Register Centre: CA Certificates. Referenced in 26.12.2006. Available from:

http://www.fineid.fi/vrk/fineid/home.nsf/pages/FA842EE9BB3C7AA5C2257054002D3FA9

APPENDIX A: Population Register Centre Root CA Certificate

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 100505 (0x18899)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=FI, ST=Finland, O=Vaestorekisterikeskus CA, OU=Certification
Authority Services, OU=Varmennepalvelut, CN=VRK Gov. Root CA
        Validity
            Not Before: Jan 10 12:59:05 2003 GMT
            Not After : Jan  9 12:58:30 2019 GMT
        Subject: C=FI, ST=Finland, O=Vaestorekisterikeskus CA, OU=Valtion
kansalaisvarmenteet, CN=VRK Gov. CA for Citizen Qualified Certificates
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:b9:02:3e:76:ee:89:71:0c:73:ad:91:04:14:f8:
                    11:b5:15:4b:34:c4:ec:bc:43:0d:b3:dc:d9:a0:57:
                    bc:8e:91:84:70:8b:f0:bb:68:06:ef:6b:27:bb:d7:
                    6a:bc:00:f1:a2:0d:af:8f:7f:43:7f:5f:34:0d:ca:
                    75:96:cb:30:b6:92:bb:fe:7d:0a:b9:62:31:b2:a5:
                    16:e7:bd:c8:80:b4:66:c9:41:25:c1:a1:7e:7c:79:
                    5e:ac:77:90:96:2c:a1:8c:a3:58:07:c6:c5:cd:53:
                    a2:fd:bf:d5:e5:49:62:d4:1f:8c:5b:62:f4:2d:fa:
                    8c:5f:a6:d8:09:6c:ae:44:fb:bd:4e:60:2f:03:2f:
                    42:94:eb:19:a5:11:fb:8f:35:06:75:45:fd:e9:aa:
                    a9:44:7e:64:23:3f:6e:2e:6d:c8:32:dd:90:07:55:
                    31:41:87:ba:d3:eb:aa:70:f8:be:73:01:53:d8:04:
                    2c:94:1b:ba:dd:5c:74:bd:86:e9:51:6b:86:3c:c3:
                    70:45:44:1c:5b:0a:11:ec:73:bb:6e:2a:4f:64:42:
                    63:46:85:00:09:ec:27:49:c7:75:79:90:fb:2b:c6:
                    7b:b3:b4:eb:a7:8a:67:81:2a:80:79:5f:7c:20:be:
                    4f:5f:eb:89:fa:d0:9a:74:aa:e9:a7:63:89:7d:57:aa:19
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            Netscape Cert Type:
                SSL CA, S/MIME CA
            X509v3 Certificate Policies:
                Policy: 1.2.246.517.1.10.1.1
                  User Notice:
                    Explicit Text: Varmennepolitiikka on saatavilla - Certifikat
policy finns - Certificate policy is available http://www.fineid.fi/cps1
                  CPS: http://www.fineid.fi/cps1/

            Authority Information Access:
                CA Issuers - URI:http://proxy.fineid.fi/ca/vrkrootc.crt

            X509v3 Key Usage: critical
                Digital Signature, Non Repudiation, Certificate Sign, CRL Sign
            X509v3 Authority Key Identifier:
keyid:DB:E9:E1:9B:D2:D1:24:0B:FC:AB:E3:A0:67:EA:AE:9C:4B:77:F4:B0

            X509v3 CRL Distribution Points:
                URI:http://proxy.fineid.fi/arl/vrkroota.crl

            X509v3 Subject Key Identifier:
                88:5A:6F:1D:42:47:82:86:FD:D7:E9:0D:B2:57:CF:4D:50:28:04:17
```

```
    Signature Algorithm: sha1WithRSAEncryption
         45:e2:b7:ac:a9:40:ef:b4:45:b5:53:2b:9e:d2:29:3d:63:b2:
         a1:3c:75:48:b0:2f:ca:1e:be:f7:41:88:5a:51:e0:7c:44:65:
         9c:bc:7b:f3:86:02:f1:77:1d:cf:c7:8d:cf:1c:3a:39:6c:61:
         3a:2a:ce:d8:35:e9:c3:85:23:8b:c7:67:ec:82:f2:b5:a1:e1:
         3a:6e:5a:0b:e4:4b:cd:21:ff:f8:dc:c1:e0:1a:ca:9e:84:fd:
         9d:33:f7:6f:2f:4c:d2:0b:04:3d:f8:60:94:2f:a5:4e:2e:ee:
         3c:a1:49:a6:37:b7:3c:9b:2a:39:52:02:8e:65:6a:18:88:df:
         66:bd:30:d6:57:1a:83:6f:fa:3f:8c:2a:ad:4d:26:4a:60:a7:
         2e:bf:54:46:b9:67:84:5d:47:1e:37:fc:46:61:b3:8e:56:bf:
         14:df:11:1f:a7:50:2d:65:a1:09:e0:14:a3:92:8d:d5:86:dc:
         68:4e:02:1d:77:9c:cf:63:60:04:81:b4:2e:ce:35:d7:6f:a0:
         1c:9f:cf:05:0e:43:e0:4e:7f:4f:11:d9:bb:d9:03:ed:82:0c:
         52:3c:3e:e6:2a:c4:21:6f:04:c0:a9:41:20:9d:54:be:ad:11:
         8c:5e:58:84:1b:fa:e2:10:b1:f0:04:7e:30:b7:01:0b:93:36:
         1f:d4:89:6f
-----BEGIN CERTIFICATE-----
MIIFjDCCBHSgAwIBAgIDAYiZMA0GCSqGSIb3DQEBBQUAMIGjMQswCQYDVQQGEwJG
STEQMA4GA1UECBMHRmlubGFuZDEhMB8GA1UEChMYVmFlc3RvcmVraXN0ZXJpa2Vz
a3VzIENBMSkwJwYDVQQLEyBDZXJ0aWZpY2F0aW9uIEF1dGhvcml0eSBTZXJ2aWNl
czEZMBcGA1UECxMQVmFybWVubmVwYWx2ZWx1dDEZMBcGA1UEAxMQVlJLIEdvdi4g
Um9vdCBDQTAeFw0wMzAxMTAxMjU5MDVaFw0xOTAxMDkxMjU4MzBaMIGhMQswCQYD
VQQGEwJGSTEQMA4GA1UECBMHRmlubGFuZDEhMB8GA1UEChMYVmFlc3RvcmVraXN0
ZXJpa2Vza3VzIENBMSQwIgYDVQQLExtWYWx0aW9uIGthbnNhbGFpc3Zhcm1lbnRl
ZXQxNzA1BgNVBAMTLlZSSyBHb3YuIENBIGZvciBDaXRpemVuIFF1YWxpZmllZCBD
ZXJ0aWZpY2F0ZXMwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC5Aj52
7olxDHOtkQQU+BG1FUs0xOy8Qw2z3NmgV7yOkYRwi/C7aAbvaye712q8APGiDa+P
f0N/XzQNynWWyzC2krv+fQq5YjGypRbnvciAtGbJQSXBoX58eV6sd5CWLKGMo1gH
xsXNU6L9v9XlSWLUH4xbYvQt+oxfptgJbK5E+71OYC8DL0KU6xmlEfuPNQZ1Rf3p
qqlEfmQjP24ubcgy3ZAHVTFBh7rT66pw+L5zAVPYBCyUG7rdXHS9hulRa4Y8w3BF
RBxbChHsc7tuKk9kQmNGhQAJ7CdJx3V5kPsrxnuztOunimeBKoB5X3wgvk9f64n6
0Jp0qumnY4l9V6oZAgMBAAGjggHHMIIBwzASBgNVHRMBAf8ECDAGAQH/AgEAMBEG
CWCGSAGG+EIBAQQEAwIBBjCBywYDVR0gBIHDMIHAMIG9BgkqgXaEBQEKAQEwga8w
gYQGCCsGAQUFBwICMHgadlZhcm1lbm5lcG9saXRpaWtrYSBvbiBzYWF0YXZpbGxh
IC0gQ2VydGlmaWthdCBwb2xpY3kgZmlubnMgLSBDZXJ0aWZpY2F0ZSBwb2xpY3kg
aXMgYXZhaWxhYmxlIGh0dHA6Ly93d3cuZmluZWlkLmZpL2NwczEwJgYIKwYBBQUH
AgEWGmh0dHA6Ly93d3cuZmluZWlkLmZpL2NwczEvMEIGCCsGAQUFBwEBBDYwNDAy
BggrBgEFBQcwAoYmaHR0cDovL3Byb3h5LmZpbmVpZC5maS9jYS92cmtyb290Yy5j
cnQwDgYDVR0PAQH/BAQDAgHGMB8GA1UdIwQYMBaAFNvp4ZvS0SQL/KvjoGfqrpxL
d/SwMDgGA1UdHwQxMC8wLaAroCmGJ2h0dHA6Ly9wcm94eS5maW5laWQuZmkvYXJs
L3Zya3Jvb3RhLmNybDAdBgNVHQ4EFgQUiFpvHUJHgob91+kNslfPTVAoBBcwDQYJ
KoZIhvcNAQEFBQADggEBAEXit6ypQO+0RbVTK57SKT1jsqE8dUiwL8oevvdBiFpR
4HxEZZy8e/OGAvF3Hc/Hjc8cOjlsYToqztg16cOFI4vHZ+yC8rWh4TpuWgvkS80h
//jcweAayp6E/Z0z928vTNILBD34YJQvpU4u7jyhSaY3tzybKjlSAo5lahiI32a9
MNZXGoNv+j+MKq1NJkpgpy6/VEa5Z4RdRx43/EZhs45WvxTfER+nUC1loQngFKOS
jdWG3GhOAh13nM9jYASBtC7ONddvoByfzwUOQ+BOf08R2bvZA+2CDFI8PuYqxCFv
BMCpQSCdVL6tEYxeWIQb+uIQsfAEfjC3AQuTNh/UiW8=
-----END CERTIFICATE-----
```

APPENDIX B: Server Certificate

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            d7:63:ed:8f:10:94:e0:db
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=FI, ST=Kuopio, L=Kuopio, O=University of Kuopio, OU=Department
of computer science CA, CN=CA/emailAddress=kontio@hytti.uku.fi
        Validity
            Not Before: Nov 17 11:53:22 2006 GMT
            Not After : Dec 17 11:53:22 2006 GMT
        Subject: C=FI, ST=Kuopio, L=Kuopio, O=University of Kuopio,
OU=Department of computer science CA, CN=CA/emailAddress=kontio@hytti.uku.fi
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:b5:f8:71:08:82:b8:a6:60:c9:f2:95:93:5d:bf:
                    4e:fd:4c:22:f1:5e:1b:3a:a8:db:74:49:a7:96:82:
                    6e:d2:62:a7:59:94:b8:00:3a:a2:af:f5:4b:7a:17:
                    3b:d1:6c:99:c0:ab:71:b5:70:03:1e:fd:1d:83:6a:
                    46:cf:c6:47:a8:f2:39:af:5a:cf:01:b5:3e:93:62:
                    a2:a8:22:27:b5:d5:20:2a:ae:85:e5:a5:29:01:76:
                    a7:fe:41:e6:fd:7a:4b:36:33:79:75:b6:2f:21:87:
                    34:fc:03:a3:75:2d:01:c2:3a:5e:98:b2:c8:33:5c:
                    30:cb:61:74:61:bf:c2:6d:c1
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                C1:8A:96:FE:C9:74:F0:FB:C2:7E:C1:45:37:7F:6C:94:A3:18:BD:20
            X509v3 Authority Key Identifier:

keyid:C1:8A:96:FE:C9:74:F0:FB:C2:7E:C1:45:37:7F:6C:94:A3:18:BD:20
                DirName:/C=FI/ST=Kuopio/L=Kuopio/O=University of
Kuopio/OU=Department of computer science
CA/CN=CA/emailAddress=kontio@hytti.uku.fi
                serial:D7:63:ED:8F:10:94:E0:DB

            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: sha1WithRSAEncryption
        64:d2:40:0b:42:30:b5:a9:6b:48:08:8e:23:55:c3:09:43:ee:
        d5:ba:17:28:ed:d5:8f:ce:4a:10:4c:98:95:03:d7:d4:c8:a0:
        87:14:3b:67:f2:6c:07:56:e7:61:54:e9:4a:86:56:77:05:a1:
        73:27:6e:6f:f1:62:14:08:c0:18:73:07:a7:9d:ff:3e:08:93:
        30:34:e5:46:6e:2b:14:2b:11:1f:b1:c9:4f:c1:07:77:fd:54:
        cc:ea:16:50:bb:d0:ee:56:a6:52:7e:8b:51:d7:ed:3c:0a:14:
        12:4a:57:66:85:b2:8d:21:3e:d4:ea:2f:e8:de:3a:f1:01:d3:53:c9
-----BEGIN CERTIFICATE-----
MIID7DCCA1WgAwIBAgIJANdj7Y8QlODbMA0GCSqGSIb3DQEBBQUAMIGrMQswCQYD
VQQGEwJGSTEPMA0GA1UECBMGS3VvcGlvMQ8wDQYDVQQHEwZLdW9waW8xHTAbBgNV
BAoTFFVuaXZlcnNpdHkgb2YgS3VvcGlvMSowKAYDVQQLEyFEZXBhcnRtZW50IG9m
IGNvbXB1dGVyIHNjaWVuY2UgQ0ExCzAJBgNVBAMTAkNBMSIwIAYJKoZIhvcNAQkB
FhNrb250aW9AaHl0dGkudWt1LmZpMB4XDTA2MTExNzExNTMyMloXDTA2MTIxNzEx
NTMyMlowgasxCzAJBgNVBAYTAkZJMQ8wDQYDVQQIEwZLdW9waW8xDzANBgNVBAcT
Bkt1b3BpbzEdMBsGA1UEChMUVW5pdmVyc2l0eSBvZiBLdW9waW8xKjAoBgNVBAsT
IURlcGFydG1lbnQgb2YgY29tcHV0ZXIgc2NpZW5jZSBDQTELMAkGA1UEAxMCQ0Ex
IjAgBgkqhkiG9w0BCQEWE2tvbnRpb0BoeXR0aS51a3UuZmkwgZ8wDQYJKoZIhvcN
AQEBBQADgY0AMIGJAoGBALX4cQiCuKZgyfKVk12/Tv1MIvFeGzqo23RJp5aCbtJi
```

```
p1mUuAA6oq/1S3oXO9FsmcCrcbVwAx79HYNqRs/GR6jyOa9azwG1PpNioqgiJ7XV
ICquheWlKQF2p/5B5v16SzYzeXW2LyGHNPwDo3UtAcI6XpiyyDNcMMthdGG/wm3B
AgMBAAGjggEUMIIBEDAdBgNVHQ4EFgQUwYqW/sl08PvCfsFFN39slKMYvSAwgeAG
A1UdIwSB2DCB1YAUwYqW/sl08PvCfsFFN39slKMYvSChgbGkga4wgasxCzAJBgNV
BAYTAkZJMQ8wDQYDVQQIEwZLdW9waW8xDzANBgNVBAcTBkt1b3BpbzEdMBsGA1UE
ChMUVW5pdmVyc2l0eSBvZiBLdW9waW8xKjAoBgNVBAsTIURlcGFydG1lbnQgb2Yg
Y29tcHV0ZXIgc2NpZW5jZSBDQTELMAkGA1UEAxMCQ0ExIjAgBgkqhkiG9w0BCQEW
E2tvbnRpb0BoeXR0aS51a3UuZmmCCQDXY+2PEJTg2zAMBgNVHRMEBTADAQH/MA0G
CSqGSIb3DQEBBQUAA4GBAGTSQAtCMLWpa0gIjiNVwwlD7tW6Fyjt1Y/OShBMmJUD
19TIoIcUO2fybAdW52FU6UqGVncFoXMnbm/xYhQIwBhzB6ed/z4IkzA05UZuKxQr
ER+xyU/BB3f9VMzqFlC70O5WplJ+i1HX7TwKFBJKV2aFso0hPtTqL+jeOvEB01PJ
-----END CERTIFICATE-----
```

APPENDIX C: Apache configuration file *httpd_ssl.conf*

```
## SSL Virtual Host Context
##

#Default 443 -> 8443
<VirtualHost _default_:8443>

#   General setup for the virtual host
DocumentRoot "c:/Apache2/htdocs"
ServerName localhost:443
ServerAdmin kontio@hytti.uku.fi
ErrorLog c:/Apache2/logs/error_log
TransferLog c:/Apache2/logs/access_log

#   SSL Engine Switch
#   Enable/Disable SSL for this virtual host.
SSLEngine on

#   SSL Cipher Suite:
#   List the ciphers that the client is permitted to negotiate.
#   See the mod_ssl documentation for a complete list.
#SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL


#   Server Certificate:
#   Point SSLCertificateFile at a PEM encoded certificate.  If
#   the certificate is encrypted, then you will be prompted for a
#   pass phrase.  Note that a kill -HUP will prompt again.  Keep
#   in mind that if you have both an RSA and a DSA certificate you
#   can configure both in parallel (to also allow the use of DSA
#   ciphers, etc.)

SSLCertificateFile c:/Apache2/conf/ssl.crt/server.crt

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)

SSLCertificateKeyFile c:/Apache2/conf/ssl.crt/server.key

#   Server Certificate Chain:
#   Point SSLCertificateChainFile at a file containing the
#   concatenation of PEM encoded CA certificates which form the
#   certificate chain for the server certificate. Alternatively
#   the referenced file can be the same as SSLCertificateFile
#   when the CA certificates are directly appended to the server
#   certificate for convenience.

SSLCertificateChainFile c:/Apache2/conf/ssl.crt/vrkcqc.crt


#   Certificate Authority (CA):
#   Set the CA certificate verification path where to find CA
#   certificates for client authentication or alternatively one
#   huge file containing all of them (file must be PEM encoded)
#   Note: Inside SSLCACertificatePath you need hash symlinks
#         to point to the certificate files. Use the provided
#         Makefile to update the hash symlinks after changes.
```

```
SSLCACertificateFile c:/Apache2/conf/ssl.crt/vrkrootc.crt

#   Certificate Revocation Lists (CRL):
#   Set the CA revocation path where to find CA CRLs for client
#   authentication or alternatively one huge file containing all
#   of them (file must be PEM encoded)
#   Note: Inside SSLCARevocationPath you need hash symlinks
#         to point to the certificate files. Use the provided
#         Makefile to update the hash symlinks after changes.

SSLCARevocationFile c:/apache2/conf/ssl.crl/list.crl

#   Client Authentication (Type):
#   Client certificate verification type and depth.  Types are
#   none, optional, require and optional_no_ca.  Depth is a
#   number which specifies how deeply to verify the certificate
#   issuer chain before deciding the certificate is not valid.

# Directory cgi-bin/secure (under the script execution directory) requires
# strong client authentication
<Location /cgi-bin/secure>
SSLVerifyClient require
SSLVerifyDepth 2
</Location>
```

APPENDIX D: PHP configuration file *php.ini*, Windows extensions section

```
; Windows Extensions
; Note that ODBC support is built in, so no dll is needed for it.
; Note that many DLL files are located in the extensions/ (PHP 4) ext/ (PHP 5)
; extension folders as well as the separate PECL DLL download (PHP 5).
; Be sure to appropriately set the extension_dir directive.
;extension=php_mbstring.dll
;extension=php_bz2.dll
;extension=php_curl.dll
;extension=php_dba.dll
;extension=php_dbase.dll
;extension=php_exif.dll
;extension=php_fdf.dll
;extension=php_filepro.dll
;extension=php_gd2.dll
;extension=php_gettext.dll
;extension=php_ifx.dll
;extension=php_imap.dll
;extension=php_interbase.dll
extension=php_ldap.dll
;extension=php_mcrypt.dll
;extension=php_mhash.dll
;extension=php_mime_magic.dll
;extension=php_ming.dll
;extension=php_mssql.dll
;extension=php_msql.dll
;extension=php_mysql.dll
;extension=php_oci8.dll
extension=php_openssl.dll
;extension=php_oracle.dll
;extension=php_pgsql.dll
;extension=php_shmop.dll
;extension=php_snmp.dll
;extension=php_sockets.dll
;extension=php_sqlite.dll
;extension=php_sybase_ct.dll
;extension=php_tidy.dll
;extension=php_xmlrpc.dll
;extension=php_xsl.dll
```

APPENDIX E: SSL related information from Tomi Kontio's PRC Citizen Qualified Certificate

SSL_VERSION_INTERFACE	mod_ssl/2.2.3
SSL_VERSION_LIBRARY	OpenSSL/0.9.8b
SSL_PROTOCOL	SSLv3
SSL_COMPRESS_METHOD	NULL
SSL_CIPHER	RC4-MD5
SSL_CIPHER_EXPORT	false
SSL_CIPHER_USEKEYSIZE	128
SSL_CIPHER_ALGKEYSIZE	128
SSL_CLIENT_VERIFY	SUCCESS
SSL_CLIENT_M_VERSION	3
SSL_CLIENT_M_SERIAL	3B9F1EE2
SSL_CLIENT_V_START	Jul 18 16:53:43 2006 GMT
SSL_CLIENT_V_END	Jul 17 21:59:59 2011 GMT
SSL_CLIENT_V_REMAIN	1689
SSL_CLIENT_S_DN	/C=FI/serialNumber=14416500C/GN=TOMI/SN=KONTIO/CN=KONTIO TOMI 14416500C
SSL_CLIENT_S_DN_C	FI
SSL_CLIENT_S_DN_CN	KONTIO TOMI 14416500C
SSL_CLIENT_S_DN_G	TOMI
SSL_CLIENT_S_DN_S	KONTIO
SSL_CLIENT_I_DN	/C=FI/ST=Finland/O=Vaestorekisterikeskus CA/OU=Valtion kansalaisvarmenteet/CN=VRK Gov. CA for Citizen Qualified Certificates
SSL_CLIENT_I_DN_C	FI
SSL_CLIENT_I_DN_ST	Finland
SSL_CLIENT_I_DN_O	Vaestorekisterikeskus CA
SSL_CLIENT_I_DN_OU	Valtion kansalaisvarmenteet
SSL_CLIENT_I_DN_CN	VRK Gov. CA for Citizen Qualified Certificates
SSL_CLIENT_A_KEY	rsaEncryption
SSL_CLIENT_A_SIG	sha1WithRSAEncryption
SSL_SERVER_M_VERSION	1
SSL_SERVER_M_SERIAL	01
SSL_SERVER_V_START	Nov 17 12:51:45 2006 GMT
SSL_SERVER_V_END	Nov 14 12:51:45 2016 GMT
SSL_SERVER_S_DN	/C=FI/ST=Kuopio/L=Kuopio/O=University of Kuopio/OU=Department of computer science/CN=localhost/emailAddress=kontio@hytti.uku.fi
SSL_SERVER_S_DN_C	FI
SSL_SERVER_S_DN_ST	Kuopio
SSL_SERVER_S_DN_L	Kuopio
SSL_SERVER_S_DN_O	University of Kuopio
SSL_SERVER_S_DN_OU	Department of computer science
SSL_SERVER_S_DN_CN	localhost
SSL_SERVER_S_DN_Email	kontio@hytti.uku.fi
SSL_SERVER_I_DN	/C=FI/ST=Kuopio/L=Kuopio/O=University of Kuopio/OU=Department of computer science CA/CN=CA/emailAddress=kontio@hytti.uku.fi
SSL_SERVER_I_DN_C	FI
SSL_SERVER_I_DN_ST	Kuopio
SSL_SERVER_I_DN_L	Kuopio
SSL_SERVER_I_DN_O	University of Kuopio
SSL_SERVER_I_DN_OU	Department of computer science CA
SSL_SERVER_I_DN_CN	CA
SSL_SERVER_I_DN_Email	kontio@hytti.uku.fi
SSL_SERVER_A_KEY	rsaEncryption
SSL_SERVER_A_SIG	sha1WithRSAEncryption

SSL_SESSION_ID	3CA0737D1F151E7FCBFC7E5E7AA0246F080962B8758A1C1C31E8E27BEC5667C1
SSL_SERVER_CERT	same as in Appendix B
SSL_CLIENT_CERT	
SSL_CLIENT_CERT_CHAIN	

APPENDIX F: Certificate Revocation List updating script *crl.php*

```
<?php

//Very simple main program to start the operation
if ($ldap_info = fetch_list()){
     install_list($ldap_info['binary']);
}

function fetch_list(){
$ldap['connection'] = ldap_connect("193.229.0.210", 389); //tai alternatively by
IP address $connection = ldap_connect("ldap.fineid.fi", 389);

//Anonymous bind to the LDAP server
if(@ldap_bind($ldap['connection'])){
     /*Execute the query. Search parameters searchBase = "dmdName=fineid, c=fi"
     can be found from PRC's documentation*/
     $ldap['result'] = ldap_search($ldap['connection'], "dmdName=fineid, c=fi",
     "cn=VRK Gov. CA for Citizen Qualified Certificates",
     array("certificaterevocationList"));
     //Get the first (and only) record from the result
     $ldap['record'] = ldap_first_entry($ldap['connection'], $ldap['result']);
     /*By using the PHP function ldap_get_value_len we read the value in binary
     form. If the binary form wouldn't be used the reading would fail.
     The value must include some sort of "end of result" marker which is
     encountered in non-binary form. */
     $ldap['binary'] = ldap_get_values_len($ldap['connection'], $ldap['record'],
     'certificaterevocationlist');
     ldap_close($ldap['connection']);
}
else
{
     print 'Connection to the LDAP-server failed.';
     $ldap = false;
}
     return $ldap;
}


function install_list($binary_list){
     //Open the new file for the CRL list
     if($CRL_file = fopen('newlist.crl', 'w'))
     {
          /*Define constants to be used as parameters
          Notice that all the file system locations might be different in all
          systems */
          $openssl_location = '/openssl/bin/openssl.exe ';
          $openssl_parameters = 'crl -in newlist.crl -inform DER -outform PEM -
          out list.crl';
          $new_list = 'c:/apache2/conf/ssl.crl/list.crl';
          $old_list = 'c:/apache2/conf/ssl.crl/old.crl';

          //Write the fetched binary infromation to the file
          fwrite ($CRL_file, $binary_list[0]);
          //Close the file
          fclose($CRL_file);
          /*Execute the external system command from PHP-script.
          The command changes the coding of the CRL list from DER to PEM
          encoding*/
          exec($openssl_location.' '.$openssl_parameters);

          //Delete the old copy of CRL list
```

```
        unlink($old_list);

        /*Rename the old CRL list. There are two reasons for this:
        1. If something goes wrong the server won't be left without CRL list
        2. The new list cannot be given the same name as the existing list
        has*/
        if (@rename($new_list, $old_list)){
             if (@rename('list.crl', $new_list))
             {
                  print 'Installing and downloading the CRL list was '.
                  'successful. Restarting Apache...';
             }
             else
             {
                  rename($old_list, $new_list);
                  print 'Installing and downloading the CRL list failed! '.
                  'Using the old CRL list';
             }
        }
        else
         {
             print 'Creation of the backup CRL failed';
        }
    }
    else
    {
        print 'Unable to open the file to save the CRL';
    }
}
?>
```