



Configuring two-way SSL using Estonian eID smartcards in Ubuntu Apache2 webserver

Document information	
Creation time	06.02.2019
Client	RIA
Author	Urmas Vanem, OctoX
Version	23.12/2

Version information		
Date	Version	Changes/notes
06.02.2019	19.02/1	Public version.
20.02.2019	19.02/1	Added chapter <i>additional configuration options</i> : firewall and OCSP configuration, default website removal. Changed by Urmas Vanem.
12.12.2019	19.12/1	Added recommendations for securing Apache. Changed by Urmas Vanem.
16.12.2020	20.12/1	Added requirement for end-user certificate to have correct <code>extendedKeyUsage</code> field and "right" certificate issuer. See chapter "(Additional) filtering of user certificate". Changed by Urmas Vanem
17.12.2020	20.12/2	Added directive <code>SSLCADNRequestPath</code> , see chapter "Filtering CA certificates for clients". Changed by Urmas Vanem
13.01.2021	21.01/1	Added demonstrative configuration file. Added HSTS configuration. Changed by Urmas Vanem
21.01.2021	21.01/2	<code>SSLCSPEnable</code> directive replaced from <code>on</code> to <code>leaf</code> . Updated TLS 1.2 cipher recommendations. Updated TLS version usage recommendations. Variable names in <code>Democonf</code> and document are synchronized. Changed by Urmas Vanem
27.01.2021	21.01/3	Added "mobiil-id" filter. Changed by Urmas Vanem

Ubuntu/Apache2 SSL configuration



Simple guide, Estonian eID view

26.02.2021	21.02/1	Added alternative possibility to filter intermediate certificate authorities supported for authentication using SSLCADNRequestFile directive. Changed by Urmas Vanem
27.04.2021	21.04/1	Support for aged ESTEID-SK 2011 certificates removed. Changed by Urmas Vanem
25.11.2021	21.11/1	Ubuntu version updated to Ubuntu Server 21.10. Apache version updated to 2.4.48. Added guidance for ECC certificates. Updated TLS and cipher recommendations.
21.02.2023	23.02/1	Ubuntu version updated to Ubuntu Server 22.04. Apache version updated to 2.4.55. Updates in virtualhost configuration. Changed by Urmas Vanem
27.12.2023	23.12/1	Removed ESTEID-SK 2015 chain. Changed by Urmas Vanem
27.12.2023	23.12/2	Removed old OCSP responder certificate.



Introduction

In this guide we describe:

- How to install and configure Apache2 (v. 2.4.55) webserver on Ubuntu Server 22.04!?
- How to configure one-way SSL with Apache webserver.
- How to configure two-way SSL with Apache webserver using Estonian eID cards.
- In addition, we give some security recommendations and check some useful configuration settings.

Apache2 installation and configuration

Installation

1. Renew Ubuntu package data, in terminal run „apt update“.



```
root@ubuntu2204:~# apt update
Hit:1 http://ee.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ee.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ee.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://ee.archive.ubuntu.com/ubuntu jammy-security InRelease
Get:5 http://ee.archive.ubuntu.com/ubuntu jammy/main amd64 DEP-11 Metadata [423 kB]
Get:6 http://ee.archive.ubuntu.com/ubuntu jammy/main DEP-11 48x48 Icons [100.0 kB]
Get:7 http://ee.archive.ubuntu.com/ubuntu jammy/main DEP-11 64x64 Icons [148 kB]
Get:8 http://ee.archive.ubuntu.com/ubuntu jammy/universe amd64 DEP-11 Metadata [
```

Picture 1 – updating packages

2. Install Apache2-e, in terminal run „apt install apache2“.



```
root@ubuntu2204:~# apt install apache2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2-data apache2-utils mailcap mime-support ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
The following NEW packages will be installed:
  apache2 apache2-data apache2-utils mailcap mime-support ssl-cert
0 upgraded, 6 newly installed, 0 to remove and 64 not upgraded.
```

Picture 2 - Apache2 installation

As result of previous steps Apache server is installed now¹:

¹ Today (21.02.2023) the latest version of Apache is 2.4.55. Version 2.4.52 is included with Ubuntu by default.

Ubuntu/Apache2 SSL configuration



Simple guide, Estonian eID view

```
root@ubuntu2204:~# apache2 -v
Server version: Apache/2.4.52 (Ubuntu)
Server built:   2023-01-23T18:34:42
root@ubuntu2204:~#
```

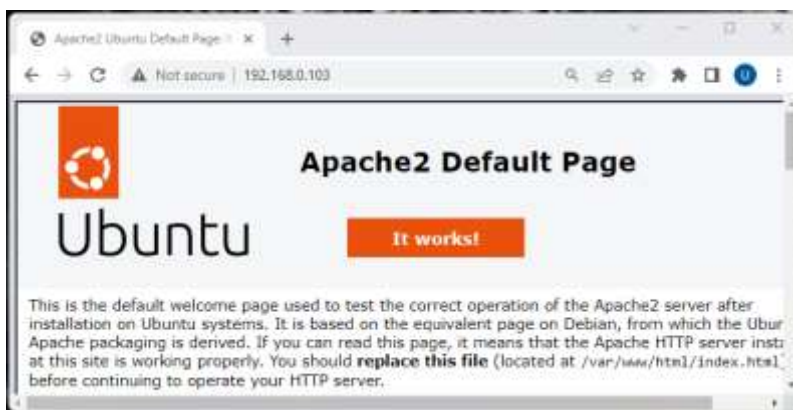
Picture 3 – querying Apache version

Let's upgrade Apache to version 2.4.55 now. There are many instructions for this on the internet, and we will not describe this procedure here².

```
root@ubuntu2204:~# apache2 -v
Server version: Apache/2.4.55 (Ubuntu)
Server built:   2023-01-19T19:55:31
root@ubuntu2204:~#
```

Picture 4 – updated Apache version is 2.4.55

Apache2 web server with 4version 2.4.55 now runs in insecure http mode:



Picture 5 - Apache webserver in default configuration

Configuration

Enabling one-way SSL

Enable SSL for Apache2, in terminal run „a2enmod ssl“ and restart Apache2 service.

² <https://ubiq.co/tech-blog/upgrade-apache-version-ubuntu/> for example.



```
root@ubuntu2204:~# a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create s
elf-signed certificates.
To activate the new configuration, you need to run:
systemctl restart apache2
root@ubuntu2204:~# systemctl restart apache2
root@ubuntu2204:~#
```

Picture 6 – Enable SSL and restart service

Creating private key certificate request file

ECC

At first, we create private key based on ECC algorithm:

1. „openssl ecparam -name secp384r1 -genkey -noout -out Apache2204.key“

And then we create certificate request file based on just created private key:

2. „openssl req -new -key Apache2204.key -out Apache2204.csr -subj /C=EE/O=OctoX/CN=Apache2204.octox.demo -reqexts SAN -config <(cat /etc/ssl/openssl.cnf <(printf \"[SAN]\\nsubjectAltName=DNS:Apache2204.octox.demo,DNS:MYWEBSERVER.octox.demo\"))“
3

```
uv@ubuntu2204:~/temp$ openssl ecparam -name secp384r1 -genkey -noout -out Apache
2204.key
uv@ubuntu2204:~/temp$ openssl req -new -key Apache2204.key -out Apache2204.csr -
subj /C=EE/O=OctoX/CN=Apache2204.octox.demo -reqexts SAN -config <(cat /etc/ssl/
openssl.cnf <(printf \"[SAN]\\nsubjectAltName=DNS:Apache2204.octox.demo,DNS:MYWEBS
ERVER.octox.demo\"))
uv@ubuntu2204:~/temp$
```

Picture 7 – creating private key and certificate request file

Notes about variables with yellow background:

1. Apache2204.key is private key of certificate;
2. Apache2204.csr is certificate request file for certification authority.
3. CN=Apache2204.octox.demo is *common name* for certificate.
4. DNS:Apache2204.octox.demo and DNS:MYWEBSERVER.octox.demo are SAN DNS names for certificate. These names must correspond to real website names⁴. And naturally the names must be resolvable in name services.

³ In addition to the certificate attributes C, O and CN described on the command line, it is also possible to describe the attributes L, OU and S if desired. However, only CN can also be used.

⁴ Modern browsers do not trust sites where at least one SAN DNS name is not equal to website DNS name.



Contents on certificate signing request file can be viewed by running „openssl req -in Apache2204.csr -noout -text“.

```
uv@ubuntu2204:~/temp$ openssl req -in Apache2204.csr -noout -text
Certificate Request:
Data:
  Version: 1 (0x0)
  Subject: C = EE, O = OctoX, CN = Apache2204.octox.demo
  Subject Public Key Info:
    Public Key Algorithm: id-ecPublicKey
    Public-Key: (384 bit)
    pub:
      04:6d:e0:54:c9:00:ad:a4:f5:3f:61:a5:20:07:95:
      43:bf:8d:e3:e6:cd:24:12:b6:52:90:5e:72:0b:fd:
      ea:49:da:2a:a8:2b:23:32:57:d4:96:ac:71:f3:c8:
      b3:05:db:08:b0:d8:55:38:1c:98:11:40:68:35:98:
      88:73:07:62:8f:47:cd:29:8c:ba:d8:4f:a9:80:60:
      5d:99:a1:a3:5a:2c:61:8c:a0:43:67:10:d0:a9:11:
      bf:73:fc:9e:47:4c:12
    ASN1 OID: secp384r1
    NIST CURVE: P-384
  Attributes:
    Requested Extensions:
      X509v3 Subject Alternative Name:
        DNS:Apache2204.octox.demo, DNS:MYWEBSEVER.octox.demo
  Signature Algorithm: ecdsa-with-SHA256
  Signature Value:
    30:65:02:38:4c:a0:90:03:3e:fb:49:ef:ca:68:9f:d5:b2:68:
    31:76:10:f6:d7:9a:87:2b:22:9d:7f:92:9f:f0:ff:b6:f0:26:
    77:8b:ee:96:59:40:3c:52:8a:f3:78:2c:a3:5f:80:fa:02:31:
    00:cd:fd:7b:15:bc:64:36:3f:0c:18:40:bf:b5:0d:37:4c:49:
    16:76:2c:c4:59:58:7d:48:4c:f3:42:1b:d0:f1:5e:1a:40:32:
    3c:b0:c4:ad:3f:01:f7:60:e5:a0:a2:82:1b
uv@ubuntu2204:~/temp$
```

Picture 8 - certificate signing request includes request for two SAN DNS names

RSA

The following section is left here from the previous version of the document in case somebody for any reason wants to continue using certificates based on the RSA algorithm. In the following chapters of the document, we continue with certificates based on ECC algorithm.

Now we create certificate signing request and private key, in terminal run „openssl req -newkey rsa:2048 -keyout Apache2021.key -sha256 -subj "/CN=Apache5.kaheksa.xi" -reqexts SAN -config <(cat /etc/ssl/openssl.cnf <(printf "[SAN]\nsubjectAltName=DNS:Apache2021.kaheksa.xi,DNS:Apache5.kaheksa.xi")) -out Apache2021.csr -nodes“.

```
uv@Ubuntu8: ~$ openssl req -newkey rsa:2048 -keyout Apache2021.key -sha256 -subj
"/CN=Apache5.kaheksa.xi" -reqexts SAN -config <(cat /etc/ssl/openssl.cnf <(print
f "[SAN]\nsubjectAltName=DNS:Apache2021.kaheksa.xi,DNS:Apache5.kaheksa.xi")) -ou
t Apache2021.csr -nodes
Generating a RSA private key
.....+++++
.+++++
writing new private key to 'Apache2021.key'
-----
uv@Ubuntu8:~$
```

Picture 9 – generating private key and certificate signing request

Notes about variables with yellow background:



1. Apache2021.key is certificate private key.
2. Apache2021.csr is certificate service request.
3. Apache5.kaheksa.xi is a subject name for certificate.
4. Apache2021.kaheksa.xi and Apache5.kaheksa.xi are certificate SAN DNS names. These names must correspond to real website names⁵. And naturally the names must be resolvable in name services.

Contents on certificate signing request file can be viewed by running „openssl req -in Apache2021.csr -noout -text“ in terminal.

```
uv@Ubuntu8: -
File Edit View Search Terminal Help
uv@Ubuntu8:~$ openssl req -in Apache2021.csr -noout -text
Certificate Request:
Data:
  Version: 1 (0x0)
  Subject: CN = Apache5.kaheksa.xi
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public-Key: (2048 bit)
    Modulus:
      00:c9:4f:a2:54:bd:1a:bb:88:a6:ec:16:c9:3e:28:
      ee:f6:f6:09:3a:d7:e6:8f:a6:7a:4e:57:3b:38:70:
      70:73:b0:01:95:7a:8d:c3:47:46:49:b9:12:52:20:
      08:0c:ed:f5:ec:c5:4e:25:3e:27:9b:98:67:b0:bd:
      c2:cd:00:98:54:36:d4:bf:b8:60:d9:aa:26:de:6a:
      da:11:23:2e:a9:05:94:ff:e8:bb:d2:5e:c2:68:8d:
      63:97:71:5e:0a:a0:49:fc:27:c7:28:c4:7d:53:12:
      1c:e6:2e:9d:bd:81:5b:ff:6a:e5:cf:b5:1a:1b:a3:
      5a:2e:9b:bd:0c:fe:c8:8f:ed:ff:b0:08:9a:1a:09:
      4f:88:a1:1c:c7:9d:84:53:f0:77:2f:db:ba:2a:9a:
      16:f4:78:02:ca:e2:29:f7:f0:f3:61:df:00:ce:3f:
      fa:80:c5:ca:2d:37:a4:2e:a4:8c:be:a2:b3:c9:fd:
      46:4e:20:fb:18:8b:3d:09:6a:be:01:3d:af:29:dd:
      e2:b6:63:3c:3e:46:c1:7a:9b:08:83:c9:32:c5:54:
      b2:e6:3d:a3:68:b6:8d:53:cb:36:c2:20:7d:77:63:
      c7:cf:c9:11:36:b3:47:9b:10:8f:19:06:cb:a4:0f:
      50:f5:35:bf:0d:53:82:cb:ad:3c:1f:5a:1a:2b:70:
      a4:8f
    Exponent: 65537 (0x10001)
  Attributes:
    Requested Extensions:
      X509v3 Subject Alternative Name:
        DNS:Apache2021.kaheksa.xi, DNS:Apache5.kaheksa.xi
  Signature Algorithm: sha256WithRSAEncryption
```

Picture 10 – certificate signing request includes request for two SAN DNS names

Certificate request

Certificate signing request file Apache2204.csr should be sent to certificate signer (in our demo environment it is just one test CA). As a response we get signed certificate in Base64 encoded format that should look like the following picture:

⁵ Modern browsers do not trust sites where at least one SAN DNS name is not equal to website DNS name.

Ubuntu/Apache2 SSL configuration

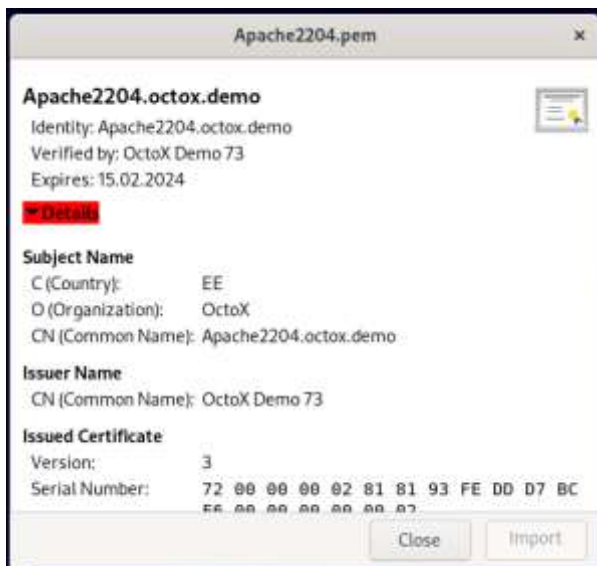


Simple guide, Estonian eID view



Picture 11 – signed certificate in text redactor

In Ubuntu the certificate looks like in the following picture:



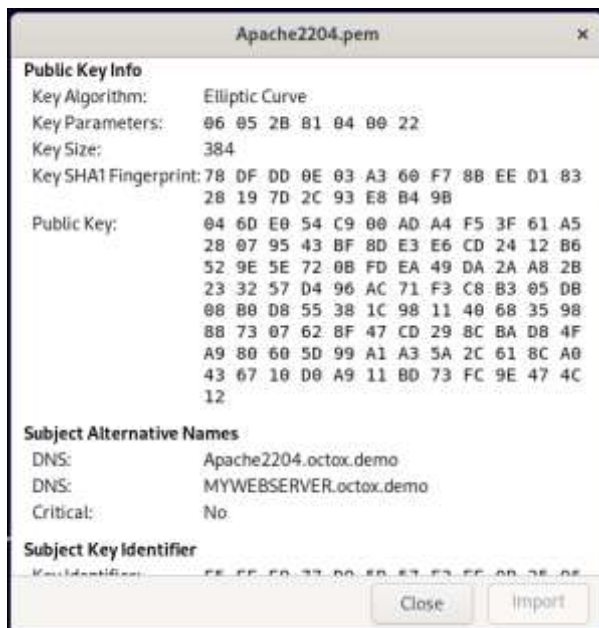
Picture 12 – ECC certificate in Ubuntu

The certificate also includes alternative SAN DNS names:

Ubuntu/Apache2 SSL configuration

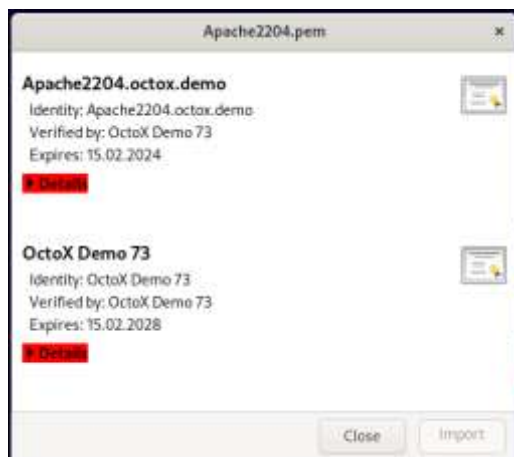


Simple guide, Estonian eID view



Picture 13 – algorithm and SAN DNS names

As we can see, the certificate issuer is a CA named „OctoX Demo 73“. Now we need to create a certificate file in which both the future web server's TLS certificate and its chain of issuers are located. To do this, we add the issuer's certificate in pem format to the certificate file of the web server in pem format and save the file named Apache2204.pem.



Picture 14 - webserver certificate and chain/issuer/root certificates in Ubuntu

The generated file must be installed in the /etc/ssl/certs folder. In addition, we need to install the private key of the web server certificate in the /etc/ssl/private folder.

Ubuntu/Apache2 SSL configuration



Simple guide, Estonian eID view



```
root@ubuntu2204: /home/uv/temp
root@ubuntu2204:/home/uv/temp# cp Apache2204.pem /etc/ssl/certs
root@ubuntu2204:/home/uv/temp# cp Apache2204.key /etc/ssl/private
root@ubuntu2204:/home/uv/temp#
```

Picture 15 – copying RCA certificate to certificates container

Now we have correctly installed all certificates and private key needed by Apache2 for one-way SSL.

Creating virtual website

For SSL configuration demonstration we create separate virtual website. At first, we create a home folder for our website `/var/www/Apache2204`.



```
root@ubuntu2204: /home/uv/temp
root@ubuntu2204:/home/uv/temp# mkdir /var/www/Apache2204
root@ubuntu2204:/home/uv/temp#
```

Picture 16 – creating website home folder

Then, for testing purposes, we put a simple webpage named `index.html` in the folder. In our example file `/var/www/html/index.html` is copied to our new folder. Then minor modifications are made in its' heading or title to understand later it is our website.

Then we prepare virtual site configuration file, in terminal run „`nano /etc/apache2/sites-available/Apache2204.conf`“.



```
root@ubuntu2204: /home/uv/temp
root@ubuntu2204:/home/uv/temp# nano /etc/apache2/sites-available/Apache2204.conf
root@ubuntu2204:/home/uv/temp#
```

Picture 17 – creating new virtual website configuration file

Now we paste the following configuration in it:

```
# Beginning of file
```

```
<Virtualhost Apache2204.octox.demo:80>
```

```
# HTTP -> HTTPS redirection
```

```
    Servername Apache2204.octox.demo
```

```
    redirect / https://Apache2204.octox.demo
```

```
</Virtualhost>
```

```
<VirtualHost Apache2204.octox.demo:443>
```

```
# General info
```

Ubuntu/Apache2 SSL configuration



Simple guide, Estonian eID view


```
ServerName Apache2204.octox.demo:443
DocumentRoot /var/www/Apache2204
```

```
# SSL configuration
    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/Apache2204.pem
    SSLCertificateKeyFile /etc/ssl/private/Apache2204.key

# Error collection configuration
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</Virtualhost>

# End of file
```

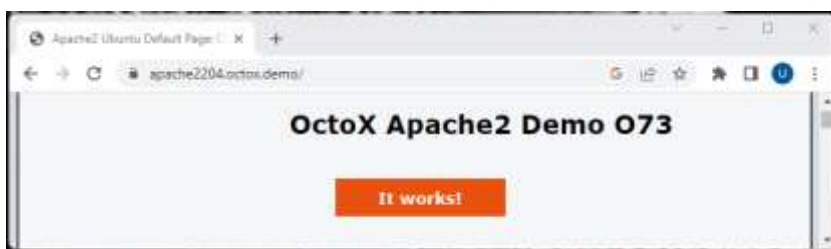
The new configuration should be activated by running „a2ensite Apache2204.conf“ in terminal. Then restart Apache2 service.



```
root@ubuntu2204:/etc# a2ensite Apache2204.conf
Enabling site Apache2204.
To activate the new configuration, you need to run:
  systemctl reload apache2
root@ubuntu2204:/etc# systemctl reload apache2
root@ubuntu2204:/etc#
```

Picture 18 – activating new virtual website and restarting Apache2

Now we have configured our new website to use one-way SSL. Also all HTTP requests to our site <http://Apache2204.octox.demo> are redirected to HTTPS site <https://Apache2204.octox.demo>.



Picture 19 - Apache web server is working and using one-way SSL!

Note. There can be many similar virtual websites with different names in same Apache2 server with single IP address.

Requiring two-way SSL

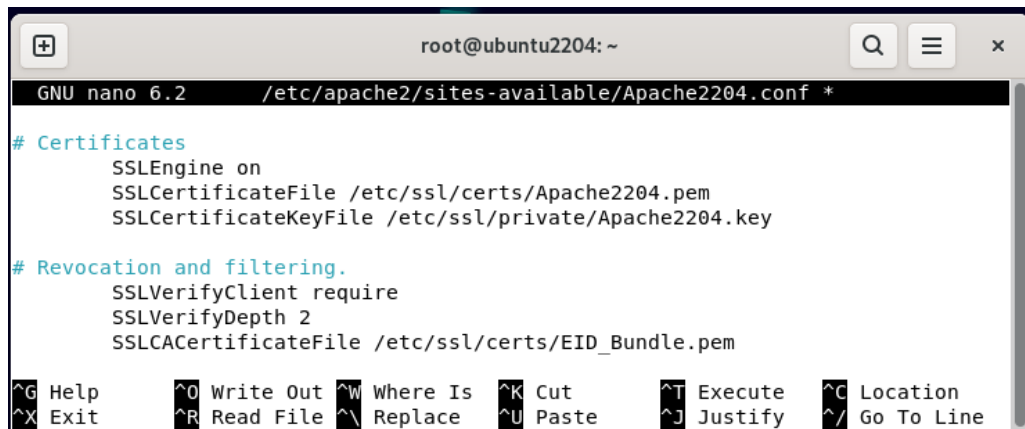
If we want to require strong Estonian eID client certificate-based authentication, we must update our configuration by adding following lines to our site configuration file Apache2204.conf:

Ubuntu/Apache2 SSL configuration



Simple guide, Estonian eID view

- SSLVerifyClient require
- SSLVerifyDepth 2
- SSLCACertificateFile /etc/ssl/certs/EID_Bundle.pem



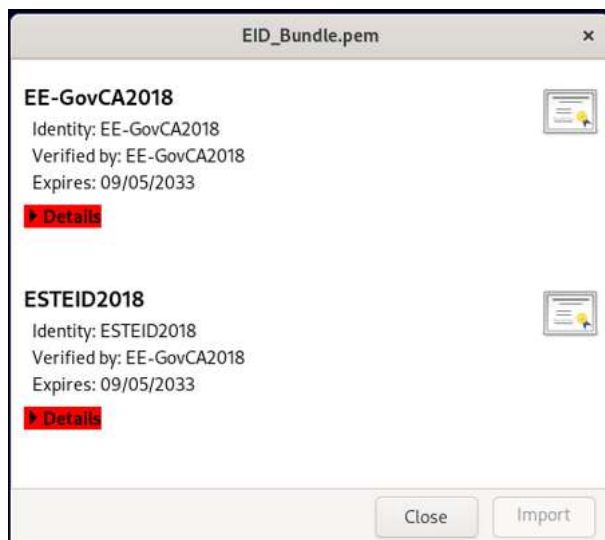
```
root@ubuntu2204: ~
GNU nano 6.2 /etc/apache2/sites-available/Apache2204.conf *
# Certificates
  SSLEngine on
  SSLCertificateFile /etc/ssl/certs/Apache2204.pem
  SSLCertificateKeyFile /etc/ssl/private/Apache2204.key

# Revocation and filtering.
  SSLVerifyClient require
  SSLVerifyDepth 2
  SSLCACertificateFile /etc/ssl/certs/EID_Bundle.pem

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/_ Go To Line
```

Picture 20 – updated configuration file, SSL section

Now we create new text file named EID_Bundle.pem⁶, which includes Estonian eID root- and intermediate certificates (EE-GovCA2018, ESTEID2018) in Base64 encrypted format. With this file we can initially filter out all client certificates supported by Apache2 web services. File opened in Ubuntu lists all four certificates we want to support:



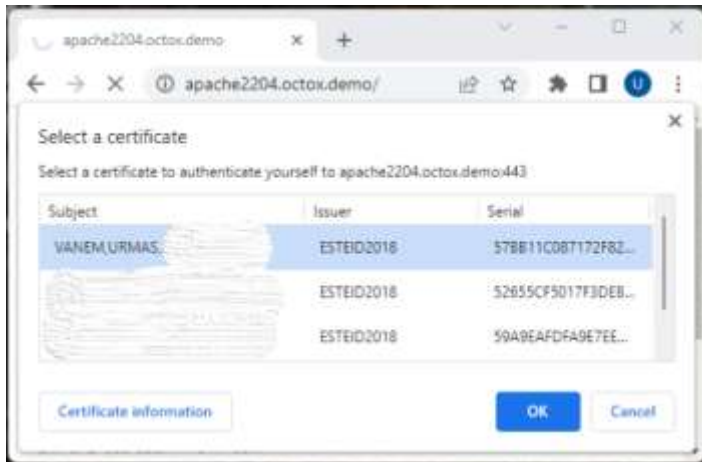
Picture 21 - supported certificates in one file

Copy file EID_Bundle.pem to folder /etc/ssl/certs and restart Apache2 webserver, in terminal run „systemctl reload apache2“.

⁶ Downloadable: https://installer.id.ee/media/id2019/EID_Bundle.pem



After accessing website Apache2204.octox.demo now, client certificate is required.



Picture 22 – client certificate request

After confirming the certificate and entering PIN we can access the website! Two-way SSL works.

Demonstrative Apache configuration file based on settings in this document, including some additional configuration options, is downloadable from address https://installer.id.ee/media/id2019/Apache_2.4.55_EID_Demo.conf.

Additional configuration options

The purpose of this document is not to give exact guidance on how to optimize or protect websites. The main purpose is to show how is possible to configure two-way SSL and use Estonian eID cards for authentication. However, in the following section we pay attention to some options which can be useful.

Firewall rules, on demand

By default, firewall is switched off on Ubuntu. To enable firewall, run “ufw enable” in terminal. When firewall is on, we probably want to create some rules to access websites on server. For Apache there are three options:

1. Apache - enables port 80
2. Apache Full – enables ports 80 and 443
3. Apache Secure – enables port 443

For creating firewall rule, run in terminal „ufw allow ‘RULE’“. For example, enabling HTTPS traffic only, run „ufw allow ‘Apache Secure’“.

Ubuntu/Apache2 SSL configuration



Simple guide, Estonian eID view

```
root@ubuntu2204: /home/uv/temp
root@ubuntu2204: /home/uv/temp# ufw enable
Firewall is active and enabled on system startup
root@ubuntu2204: /home/uv/temp# ufw allow 'Apache Secure'
Skipping adding existing rule
Skipping adding existing rule (v6)
root@ubuntu2204: /home/uv/temp#
```

Picture 23 – enabling firewall and creating Apache https rules

Running „ufw status“ in terminal shows us firewall status and active rules:

```
root@ubuntu2204: /home/uv/temp
root@ubuntu2204: /home/uv/temp# ufw status
Status: active

To Action From
--
Apache Secure ALLOW Anywhere
Apache Secure (v6) ALLOW Anywhere (v6)

root@ubuntu2204: /home/uv/temp#
```

Picture 24 – firewall is active and HTTPS is enabled

OCSP⁷

Quaranteed OCSP service

By default, configuration described above allows access to Apache2 website for all users with certificates valid in action time. Certificate revocation status is not checked! If we want to check client certificate revocation status using SK guaranteed OCSP service, we first need to make agreement with them (SK). After agreement SK allows client access to guaranteed OCSP service (with address <http://ocsp.sk.ee>) based on client IP address or certificate.

After getting access to the service, we must add following lines to our Apache2 virtual site SSL configuration section:

- `SSLCSPEnable leaf #` – enable OCSP certificate revocation check for client certificate.
- `SSLCSPEndpoint http://ocsp.sk.ee #` - determine OCSP service location.
- `SSLCSPOverrideResponder on #` - use defined OCSP responder even if OCSP responder is determined in client certificate.

Now our SSL configuration looks like presented on following picture:

⁷ Client certificate revocation check is also doable with configuring certificate revocation lists (CRL). In this document we do not deal with this because we think configuring OCSP is more reasonable.

Ubuntu/Apache2 SSL configuration



Simple guide, Estonian eID view

```
root@ubuntu2204: -
GNU nano 6.2 /etc/apache2/sites-available/Apache2204.conf *
# Certificates
  SSLEngine on
  SSLCertificateFile /etc/ssl/certs/Apache2204.pem
  SSLCertificateKeyFile /etc/ssl/private/Apache2204.key
# Revocation and filtering.
  SSLVerifyClient require
  SSLVerifyDepth 2
  SSLCACertificateFile /etc/ssl/certs/EID_Bundle.pem
# OCSP
  SSLOCSPEnable leaf
  SSLOCSPOverrideResponder on
  SSLOCSPDefaultResponder http://ocsp.sk.ee
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

Picture 25 - OCSP part is added to SSL configuration section

Reload Apache2 service with command „systemctl reload Apache2“ in terminal. Now all client certificates are checked against SK guaranteed OCSP service in authentication phase.

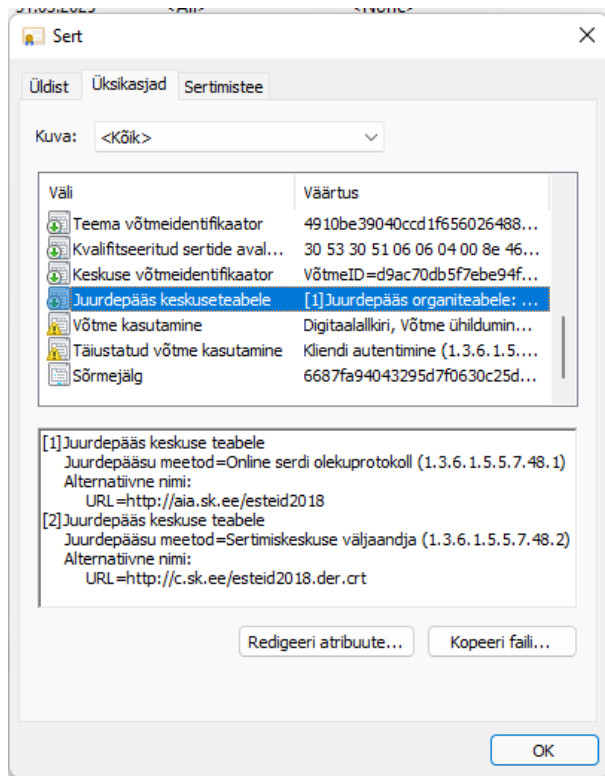
AIA OCSP service

In addition to guaranteed (payable) OCSP service, SK offers also similar but free access AIA OCSP service. AIA OCSP is basically simple OCSP service with lower availability demands and less functionality. The AIA OCSP path for eID 2018 certifications is <http://aia.sk.ee/esteid2018>.

Ubuntu/Apache2 SSL configuration



Simple guide, Estonian eID view



Picture 26 - ESTEID2018 AIA OCSP path in certificate

To enable client certificate revocation check against AIA OCSP service, we need to add following lines to our Apache2 virtual website SSL configuration section:

- SSLOCSPEnable leaf # – enable OCSP revocation check for client certificates.
- SSLOCSPUseRequestNonce off # - turn off OCSP service *nonce* requirement.



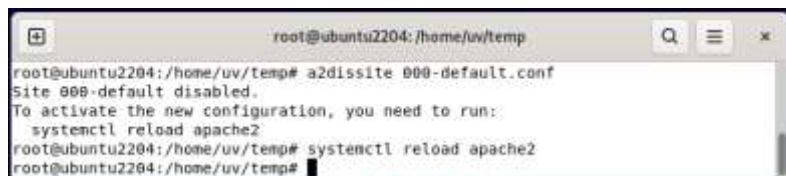
Picture 27 – AIA OCSP configuration is added to SSL section

Reload Apache2 service with command „systemctl reload Apache2“ in terminal. Now all client certificates are checked against SK AIA OCSP service determined in certificate in authentication phase.



Default website removal

After Apache2 installation we have also default website installed. To remove default website run command „a2dissite 000-default.conf“ in terminal.



```
root@ubuntu2204:/home/uv/temp
root@ubuntu2204:/home/uv/temp# a2dissite 000-default.conf
Site 000-default disabled.
To activate the new configuration, you need to run:
  systemctl reload apache2
root@ubuntu2204:/home/uv/temp# systemctl reload apache2
root@ubuntu2204:/home/uv/temp#
```

Picture 28 – default website removal

Restart Apache2 service with command „systemctl reload Apache2“ in terminal to apply new configuration.

Recommended security settings for Apache

SSL/TLS

Apache version 2.4.55 is using all SSL/TLS protocols with version higher than SSL3 by default:



```
root@ubuntu2204:/home/uv/temp
root@ubuntu2204:/home/uv/temp# grep -i -r "SSLProtocol" /etc/apache2/mods-availa
ble/
/etc/apache2/mods-available/ssl.conf:SSLProtocol all -SSLv3
root@ubuntu2204:/home/uv/temp#
```

Picture 29 - Apache version and SSL/TLS default configuration

Old unsecure TLS protocols with version number lower than TLS 1.2 should definitely no longer be used. TLS 1.2 should be the lowest version to use! For a while, we can also use the newest version TLS 1.3.

TLS 1.2 is widely supported and works fine and secure with correct configuration, but today it can be good idea to enable only TLS version 1.3. It is faster, more secure by default and needs less configuration. So, if you have no specific requirements to support TLS version 1.2, you should support TLS 1.3 protocol only! In standard situation, TLS 1.2 should be enabled only if really needed and if it is enabled, it is mandatory to allow only secure cipher suites and extensions!

To configure Apache to support only TLS protocol version 1.3, we must add following line to Apache configuration file: „SSLPROTOCOL -all +TLSv1.3“.

```
SSLPROTOCOL -all +TLSv1.3
```

Picture 30 – enabling only TLS version 1.3 in configuration file

To support TLS versions 1.2 and 1.3, add “+TLSv1.2” to configuration line.

If we want to make the change on server level, we must modify parameter SSLPROTOCOL in the file /etc/apache2/mods-available/ssl.conf.



More information about the recommendations for the use of the cipher suites can be found in the cryptographic algorithms life cycle report ordered by RIA at <https://www.id.ee/en/article/cryptographic-algorithms-life-cycle-reports-2/>.

Cipher suites

All TLS 1.3 cipher suites available today are considered to be safe, so in security point of view we can go on with no additional configuration.

It is different from TLS 1.2. There are many different TLS cipher suites available with Apache version 2.4.55.⁸ We can list available cipher suites in Apache with command „openssl ciphers -v“.

By default, only two rules are defined against ciphers:

- 1) HIGH – some ciphers with key length 128 bits and all stronger are enabled;
- 2) !aNULL – ciphers not supporting authentication are disabled.

```
SSLCipherSuite HIGH:!aNULL
```

Picture 31 – server-based configuration in file `/etc/apache2/mods-available/ssl.conf`

If we want to configure available cipher suites for example for security reason, we can use command line SSLCIPHERSUITE in Apache configuration file. Here we can use predefined aliases or exact cipher suite descriptions.

It is impossible to give an exact recommendation for configuring cipher suites because different environments have different requirements. Also, requirements and possibilities change in time. The only recommendation we can give here is to remove non-secure cipher suites from the list.

Example:

- Using following command line in configuration file: `'SSLCIPHERSUITE "ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384"'` – only described ciphers are allowed.

You can also configure cipher suites on server level by modifying parameter SSLCIPHERSUITE in file `/etc/apache2/mods-available/ssl.conf`.

More information about the recommendations for the use of the cipher suites can be found in the cryptographic algorithms life cycle report ordered by RIA at <https://www.id.ee/en/article/cryptographic-algorithms-life-cycle-reports-2/>.

⁸ We handle only TLS 1.2 ciphers in this chapter, because lower protocols should be disabled and everything is OK with TLS version 1.3 today.



SSLHONORCIPHERORDER

Important parameter related to SSLCIPHERSUITE is also SSLHONORCIPHERORDER. If this parameter has value ON, server list of cipher suites is always preferred! By default, it is not defined and default value is "off".

(Additional) filtering of user certificate

Important! To avoid access to our web service with incorrect certificates we must add following requirement:

- 1) Certificate must have correct extendedKeyUsage field;
- 2) Certificate issuer must be ESTEID2018;

So, let's add following lines to our Apache configuration:

```
<Location "/">
Require expr ( \
    %{SSL_CLIENT_I_DN_CN} == "ESTEID2018" \
and    "TLS Web Client Authentication, E-mail Protection" in PeerExtList('extendedKeyUsage') \
)
</Location>
```

The configuration above can be added to virtual host or to Apache main configuration. After enabling the configuration only certificates with correct extendedKeyUsage field issued form enabled chain can access our services.

Notes!:

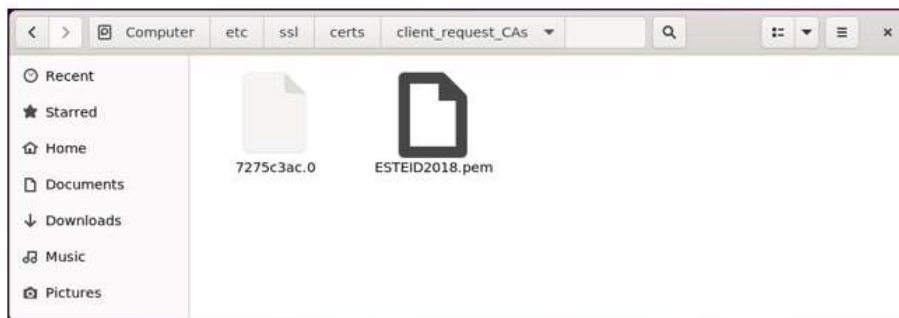
- If you have additional possibilities to filter network traffic, then implement secure configuration on that level too! SK ID Solutions (hereafter SK) published information about recommended F5 configuration in chapter "Only accept certificates with trusted key usage" in the following article: <https://github.com/SK-EID/smart-id-documentation/wiki/Secure-Implementation-Guide>
- SK recommendations for secure smart card authentication are published here in chapter "Defence: implement ID-card authentication securely": <https://github.com/SK-EID/smart-id-documentation/wiki/Secure-Implementation-Guide>
- Especially good method for filtering good and bad certificates is using OID's in end user certificates. Unfortunately, we didn't find a method to filter certificate using OID's on server level. So, if you have the possibility to do this at a web application level, please do so. Open certificate, read OID in the certificate and if it is not in trusted list, reject the authentication request! All currently known OID's are listed in chapter "Only accept certificates with trusted issuance policy" in the following article published by SK: <https://github.com/SK-EID/smart-id-documentation/wiki/Secure-Implementation-Guide>



Filtering CA certificates for clients

By default, when user tries to authenticate him- or herself with certificate to Apache web server, all user authentication certificates are listed for to, even not supported ones. To avoid listing of not supported certificates we can create supported certificate authorities list on server side. For that:

- 1) Create folder for certificates: `mkdir /etc/ssl/certs/client_request_CAs`;
- 2) Put ESTEID2018 certificate in Base64 format to folder created above;
- 3) Create certificate hash by running: `openssl rehash /etc/ssl/certs/client_request_CAs`, there is 1 new file/link in the folder now:



Picture 32 - intermediate certificates with hashes

- 4) Add directive `SSLCADNRequestPath /etc/ssl/certs/client_request_CAs` into Apache config file SSL section and save new configuration;
- 5) Restart Apache server: `systemctl reload apache2`.

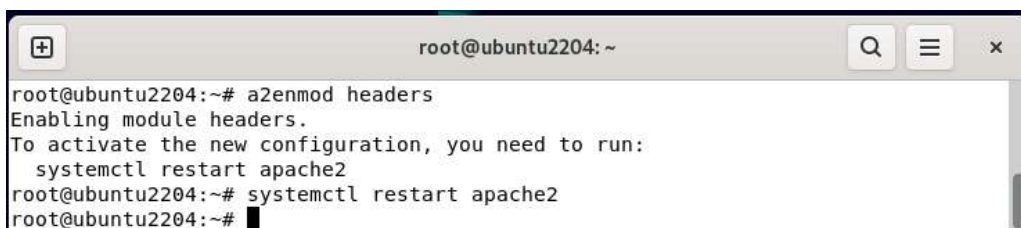
Now Apache sends information about supported certificates to client by issuing CA-s and only certificates issued by “ESTEID2018” are listed to user in authentication attempt.

Alternative directive for certificate filtering

We can replace directive `SSLCADNRequestPath` with directive `SSLCADNRequestFile`. In this case we must create file containing all supported intermediate certification authorities in pem format and describe it in our configuration file with `SSLCADNRequestFile` directive. As example, `„SSLCADNRequestFile /etc/ssl/certs/SupportedCAs.pem“` enables all intermediate authorities described in `SupportedCAs.pem` file.

Enabling HTTP Strict Transport Security (HSTS)

Enable mod-headers, run `“a2enmod headers”` in terminal:



Picture 33 – enabling Apache headers, in our example for HSTS



Add following line to Apache configuration: „Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"“.

```
# Enable HSTS.
Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"
```

Picture 34 – activating HSTS for 30 days

To apply changes restart Apache service.

Additional possibilities

In addition to TLS and cipher suite configuration there are many other things we can do to secure our server:

- Keep operating system up to date.
- Keep Apache up to date.
- Run Apache under non-root user rights.
- Disable presenting server information.
- Remove unnecessary modules.
- Disable HTTP requests.
- Add and configure *Mod Security*.
- Add and configure *Mod Evasive*.
- Disable directory listing.
- Enable logging.
- ...

Please take the list above as a short demo recommendations list. Of course, it makes sense to follow the recommendations, but there can be much more you can do to secure your server:

<https://www.google.com/search?q=how+to+secure+apache+server>.