Contents

1. Creating a key pair and certificate tax to serve as a CA certificate on the (your)server:

- create the directories to hold the CA certificate and related files:

```
sudo mkdir /etc/ssl/CA
sudo mkdir /etc/ssl/newcerts
```

- The CA needs a few additional files to operate, one to keep track of the last serial number used by the CA, each certificate must have a unique serial number, and another file to record which certificates have been issued:

```
sudo sh -c "echo '01' > /etc/ssl/CA/serial"
sudo touch /etc/ssl/CA/index.txt
```

- The third file is a CA configuration file. Though not strictly necessary, it is very convenient when issuing multiple certificates. Edit /etc/ssl/openssl.cnf, and in the [CA_default] change:

```
dir             = /etc/ssl              # Where everything is kept
database        = $dir/CA/index.txt     # database index file.
certificate     = $dir/certs/cacert.pem # The CA certificate
serial          = $dir/CA/serial        # The current serial number
private_key     = $dir/private/cakey.pem# The private key
```

- Next, create the self-signed root certificate:

```
openssl req -new -x509 -extensions v3_ca -keyout cakey.pem -out cacert.pem -days 3650
```

You will then be asked to enter the details about the certificate.

- Now install the root certificate and key:

```
sudo mv cakey.pem /etc/ssl/private/
sudo mv cacert.pem /etc/ssl/certs/
```

2. Create a key pair and certificate for the web server issued by your group's own CA (signed by the private key created for the CA certificate).

Generating a Certificate Signing Request (CSR)

```
openssl genrsa -des3 -out server.key 2048
```

Now create the insecure key, the one without a passphrase, and shuffle the key names:

```
openssl rsa -in server.key -out server.key.insecure
mv server.key server.key.secure
mv server.key.insecure server.key
```

To create the CSR, run the following command at a terminal prompt:

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

Creating a Self-Signed Certificate

To create the self-signed certificate, run the following command at a terminal prompt:

```
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

Installing the Certificate

You can install the key file server.key and certificate file server.crt, or the certificate file issued by your CA, by running following commands at a terminal prompt:

```
sudo cp server.crt /etc/ssl/certs
sudo cp server.key /etc/ssl/private
```

Enter the following to generate a certificate signed by the CA:

```
sudo openssl ca -in server.csr -config /etc/ssl/openssl.cnf
```

Then I moved the server certificate to client with the folling command:

```
scp ./server.crt gr13@172.16.0.103:/home/gr13
```

3. Configure Apache to Use SSL

Open the file with root privileges now:

```
sudo nano /etc/apache2/sites-available/default-ssl.conf
```

The entries in red were modified from the original file:

```
<IfModule mod_ssl.c>
    <VirtualHost _default_:443>
        ServerAdmin admin@grupp13.com
        ServerName grupp13.com
        ServerAlias www.grupp13.com
        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        SSLEngine on
        SSLCertificateFile /etc/apache2/ssl/server.crt
        SSLCertificateKeyFile /etc/apache2/ssl/server.key
        <FilesMatch "\.(cgi|shtml|phtml|php)$">
                        SSLOptions +StdEnvVars
        </FilesMatch>
        <Directory /usr/lib/cgi-bin>
                        SSLOptions +StdEnvVars
        </Directory>
        BrowserMatch "MSIE [2-6]" \
                        nokeepalive ssl-unclean-shutdown \
                        downgrade-1.0 force-response-1.0
        BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
    </VirtualHost>
</IfModule>
```

Save and exit the file when you are finished.

Activate the SSL Virtual Host

Now that we have configured our SSL-enabled virtual host, we need to enable it.

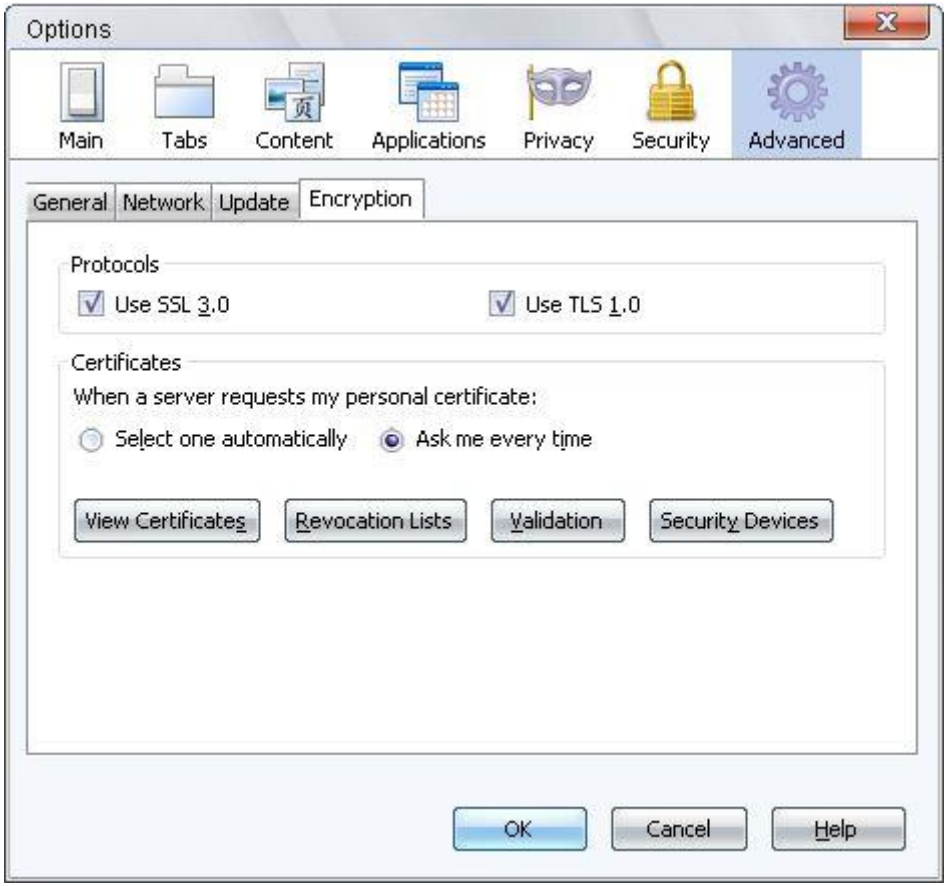
```
sudo a2ensite default-ssl.conf
```

Restart Apache

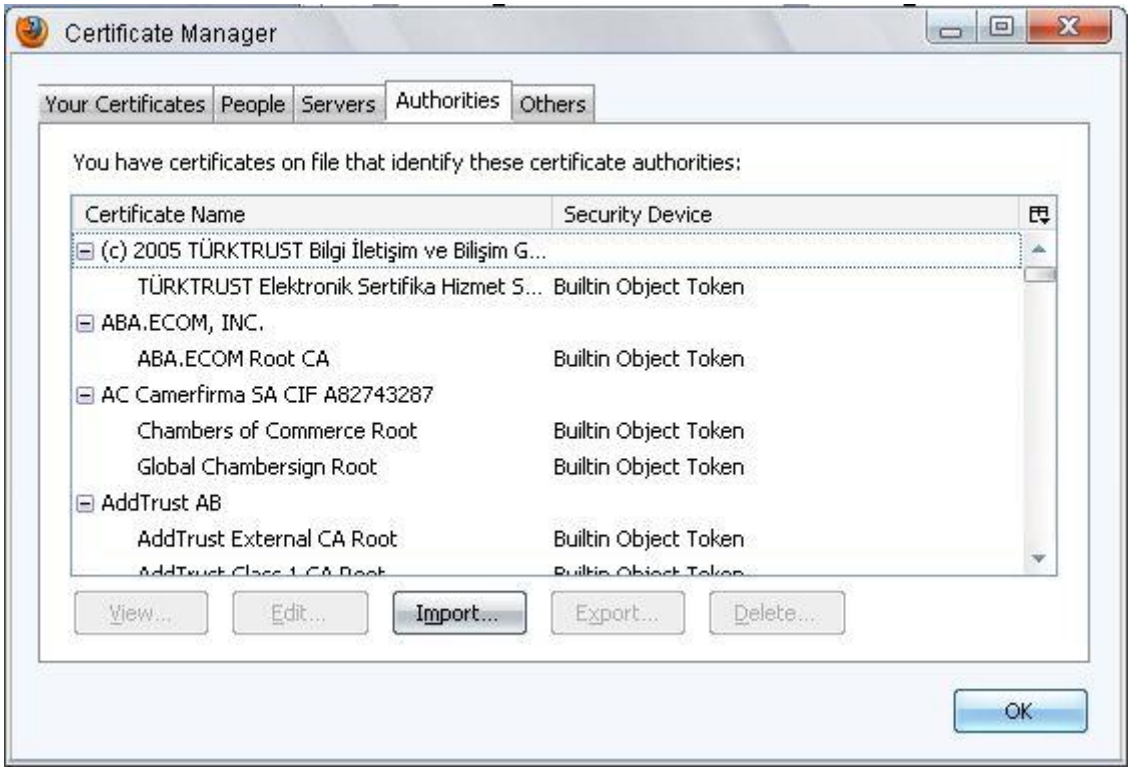
```
sudo service apache2 restart
```

4. Installing our own CA certificate in the Firefox browser:

After saving the certificate to the hard disk, then go to the *Tools* menu and select *Options...*. In the Options window, go to the *Advanced* section - *Encryption* tab and click the *"View certificates"* button.



In the *Certificate Manager* window, switch to the *Authorities* tab and click the *"Import..."* button.



Find the saved certificate file on the hard disk and click the *"Open"* button.

Select all of the checkboxes presented and click the *"OK"* button.


Downloading Certificate
You have been asked to trust a new Certificate Authority (CA).
Do you want to trust "WebMoney Transfer Root Authority" for the following purposes?
Trust this CA to identify web sites.
Trust this CA to identify email users.
Trust this CA to identify software developers.
Before trusting this CA for any purpose, you should examine its certificate and its policy and procedures (if available).
View
Examine CA certificate
OK
Cancel

5. How does HTTPS work?

   Hypertext Transfer Protocol Secure (HTTPS) which is simply your HTTP with the SSL/TLS. In other words servers and clients use the standard HTTP but over a secure SSL connection so that their communications is encrypted (with the public key) and and dencrypted (with the private key). The public key can be be distributed to anyone and only the one/ones with the corresponding private key can decrypt the information.

   1. The process starts when a client via a browser wants to reach a specific webpage (web server) which is secured with Secure Sockets Layer (SSL), so browser needs the reached server to identify itself.
   2. The reached server then will send a copy of its SSL Certificate, which includes various pieces of data and the server's public key is of those data.
   3. The certificate root will then be checked by the browser against a list of trusted CAs and that the certificate root is still valid, complete, and that its common name matchs the connected website. Once the browser accepts the certificate, so it uses the server's public key to encrypt, make, and return a symmetric session key.

   4. The server with its private key wil be able to decrypt the symmetric session and to start the encrypted session, an acknowledgement encrypted with the session key will be then send back.

   5. All transfered data will be encrypted between server and client (browser) using the session key.