KDPW

Appendix 4: using OpenSSL to obtain a certificate for A2A communication

This appendix provides instructions on how to use OpenSSL to generate a certificate signing request, which is necessary for submitting a certificate request using the "A2A Certificates" application, which will be available to KDPW participants with the implementation of the project's second stage. **A similar application will also be available to KDPW_CCP participants.** The certificate signing request should be generated directly by the applicant, within their own infrastructure, using cryptographic tools compatible with their policies. Therefore, the commands described in the attachment should be considered an example of how to use OpenSSL rather than a requirement to use this specific tool.

OBTAINING A CERTIFICATE FOR A2A COMMUNICATION

About certificates dedicated to A2A communication

An electronic certificate for A2A communication with the services provided by KDPW is issued to a given institution code based on a transmitted Certificate Signing Request (CSR). The certificate is used to establish an encrypted connection based on the TLS protocol and to authenticate the user in the relevant communication channel within A2A communications based on MQ queues.
To ensure security of the generated certificate, the private key must not leave the infrastructure of the certificate owner in the entire process of obtaining the certificate. This means that it must be created outside the KDPW infrastructure and the certificate itself should be generated in response to a transmitted Certificate Signing Request prepared in accordance with an established scheme.
In A2A communication, KDPW has established the following certificate scheme:

- Key type: RSA
- Key size: 2048
- Entity name X.509:
  - Organization Name (O=[A-Z0-9]{4,4}) - institution code, e.g. XXXX
  - Organizational Unit Name (OU=[PRD;TST]) - environment name
  - Common Name (CN= [A-Z0-9]{4,4}+"_A2A") - common name, e.g. XXXX_A2A

In A2A communication, KDPW_CCP has established the following certificate scheme:

- Key type: RSA
- Key size: 2048
- Entity name X.509:
  - Organization Name (O=[A-Z0-9]{4,4}) - institution code, e.g. XXXX
  - Organizational Unit Name (OU=[PRD;TST]) - environment name
  - Common Name (CN= [A-Z0-9]{4,4}+"_CCPA2A") - common name, e.g. XXXX_CCPA2A

The process can be carried out using a number of tools implementing cryptographic algorithms and allowing the generation of key pairs within the PKI architecture. The examples indicated in KDPW's documentation refer to OpenSSL tools, commonly available under the Apache licence; however, the actions described may also be carried out using other tools.

About OpenSSL

OpenSSL (https://www.openssl.org) is a cross-platform tool that is a set of libraries implementing basic cryptographic operations in SSL and TLS protocol support. It is distributed open source under an Apache-type licence, which means that it can be used free of charge for commercial and non-commercial purposes, subject to a number of licence conditions.
The OpenSSL installation package can be downloaded from https://wiki.openssl.org/index.php/Binaries. For the process of obtaining a certificate for A2A communication to proceed successfully, it is irrelevant on which system platform it is run.
When using OpenSSL, it is important to ensure that the appropriate cryptographic software is installed in the environment prepared to generate the Certificate Signing Request and is accessible from the directory in which the operation will be performed, and that the profile under which the procedure will be performed has the appropriate level of privileges to use it.

GENERATING A CERTIFICATE SIGNING REQUEST USING OPENSSL

Generating a CSR with interactive information input

To generate a Certificate Signing Request in a way that allows data to be entered interactively, enter the following command.

```
openssl req -newkey rsa:2048 -keyout private.key -out request.csr
```

After running the command, enter a password which protects the private key. The password will be required to access the private key for subsequent operations to be performed using it.
At a later stage of creating a Certificate Signing Request, enter additional information concerning the entity for which the certificate is to be created. It is important that the data entered into the relevant fields is consistent with the adopted scheme. Values of fields outside the scheme may be omitted, which in the case of OpenSSL means entering a dot (“.”).
For CSRs, the fields are to be filled as follows:

- Country Name (2 letter code) - „.”
- State or Province Name (full name) - „.”
- Locality Name (e.g., city) - „.”
- Organization Name (e.g., company) - institution code, e.g. XXXX
- Organizational Unit Name (e.g., section) - environment name (PRD or TST)
- Common Name (e.g., server FQDN) - common name, e.g. XXXX_A2A
- Email Address - „.”
- A challenge password - „.”
- An optional company name - „.”

Once completed, the request ready to be sent will be saved in the CSR file indicated when the command is called; in the example, this will be the “request.csr” file. The content of the CSR file should be transmitted to KDPW in the content of the request.

Automatic generation of a CSR using the full command

One of the options available when creating a Certificate Signing Request is to enter all the necessary information in the command using the -subj option available for the OpenSSL command. In this option, it is possible to provide the information required in the certificate scheme. In the case of a certificate for A2A communication within KDPW services, the data should be entered as follows.

```
-subj "/O=XXXX/OU=TST/CN=XXXX_A2A"
```

The full command to create a Certificate Signing Request has the following form.

```
openssl req -newkey rsa:2048 -subj "/O=XXXX/OU=TST/CN=XXXX_A2A" -keyout
private.key -out request.csr
```

After running the command, enter the password to secure the private key and then confirm it. As a result, the private key will be encrypted with the password, which will be required if you need to use the private key.
To enter the password directly in the command to create a CSR file, run the following command (in the example, the string “123456789” is the password).

```
openssl req -newkey rsa:2048 -subj "/O=XXXX/OU=TST/CN=XXXX_A2A" -passout
pass:123456789 -keyout private.key -out request.csr
```

Once completed, the request ready to be sent will be saved in the CSR file indicated when the command is called; in the example, this will be the “request.csr” file. The content of the CSR file should be transmitted to KDPW in the content of the request.

Extracting the content of a Certificate Signing Request

A certificate signed by the KDPW Authorisation Centre is received via dedicated application available on the Services Portal; after authentication and selecting the appropriate institution code, you can submit the application here. In the content of the application (using the text window provided for this purpose), paste the content of the generated CSR file.
To extract the contents of the file, you can use any method. Depending on your preference, you can open the CSR file in any text editor and then copy the contents to the application you are submitting. Alternatively, you can view the contents of the CSR file using the command line and the following commands:

- in Windows

```
more .\request.csr
```

- in Linux

```
cat ./request.csr
```

The displayed content of the CSR file looks similar to the example below. Importantly, when copying the content, always include the start and end tags of the request.

```
-----BEGIN CERTIFICATE REQUEST-----
MIICdTCCAV0CAQAwMDENMAsGA1UECgwEWFhYWDEMMAoGA1UECwwDVFNUMREwDwYD
VQQDDAhYWFhYX0EyQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKyl
otU7qeztCrG23DjSrJlPGdVwtj5Ikdid7BDgWTNMji+6rgYHosMtXT+slmMH3oSp
ryN9eKl1Ci3L5VdTuye7/qaPsAoHnTmdH8gSu67RVeRmqZkYfjorPQBWF+cKGhca
RMy2z0AoUfHEa51KV/lWBRo/ulm0a0V1E7sRrS/Fk/7pCJ3Vbh9KsrZIxNa4ZLux
tRYFOEoBBJ/Nri7mPOm+39hx98nR6czEOcBtGJ8KKPyZXbluZs5j1Gh7qGBO8h/h
0BM5RESMcls56qpANq21jrxT7shK1il6lbsxgGHCIJKqbzk9sPPkHYBpfeDZOb4p
L1KWU03PiLlOHoBdR88CAwEAAaAAMA0GCSqGSIb3DQEBCwUAA4IBAQAdDCLkd+LD
4MjLWdejk0L5KCC6S97M1sagfBeWBgxcv0ncfSx8laKMb9sjFWcQW75io/E5fH69
nosWQNAWdQO37vB4cRr3ihlLTrks3VqVD7OYowTEK735VYYXM9wBnhmYbY0o9SnN
UnWx/RIise1eokj9BFbW07EOZ5MiwcZ4PTVBk1AKRBHzPVNM4bOifrJskoQ8+S4g
+Jx3LTBSJ5VZBARDxKYWnkYSFV4krUa+Xlmj89G1LP3jern6j8SCJVX3tf7s+a+o
1COGvZ576NA4n1bHLfbKU4KMJiIRcpz8iW+gkJSdz1nwr00LhwVAKxjtHPMET4s+
4L1nSrXwUx24
-----END CERTIFICATE REQUEST-----
```

DOWNLOADING A GENERATED CERTIFICATE

Once a CSR has been submitted and processed by KDPW, a certificate in PEM format will be made available in the dedicated application. It can be downloaded by selecting the “Save” option available for active certificates in the view with the list of certificates issued for the institution code.
The downloaded certificate is linked with the private key generated in the creation of the CSR file. It is important not to lose the link, especially when generating a larger number of certificates. It is also possible to link a certificate to its private key in a PFX file (PKCS#12 format)
The subsequent use of a certificate depends on the internal infrastructure of the institution for which the certificate is generated.

Combining a certificate and a private key using PKCS#12 format

The PKCS #12 (PFX) format stores both the certificate and the private key in one encrypted file. To create a certificate in this format after downloading it from KDPW (PEM format), use the following command.

```
openssl pkcs12 -inkey private.key -in certificate.pem -export -out
certificate.pfx
```

If a previously created private key is protected with a password, it is necessary to enter the password to perform the operation. In addition, when creating a PFX file, it is possible to enter a password to secure the data in the file being created.

VERIFICATION

The OpenSSL commands presented in this chapter show ways to verify the correctness of the data generated. Running these operations is not necessary to obtain an A2A certificate, but it will allow you to determine the cause of any errors during the process.

Verifying the correctness of a Certificate Signing Request

```
openssl req -text -in request.csr -noout -verify
```

This command will allow to verify the signature in the CSR file, the selected algorithm and the content of the fields entered within the scheme established by KDPW.

```
Certificate request self-signature verify OK
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: O = XXXX, OU = TST, CN = XXXX_A2A
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:ac:a5:a2:d5:3b:a9:ec:ed:0a:b1:b6:dc:38:d2:
                    ac:99:4f:19:d5:70:b6:3e:48:91:d8:9d:ec:10:e0:
                    59:33:4c:8e:2f:ba:ae:06:07:a2:c3:2d:5d:3f:ac:
                    96:63:07:de:84:a9:af:23:7d:78:a9:75:0a:2d:cb:
                    e5:57:53:bb:27:bb:fe:a6:8f:b0:0a:07:9d:39:9d:
                    1f:c8:12:bb:ae:d1:55:e4:66:a9:99:18:7e:3a:2b:
                    3d:00:56:17:e7:0a:1a:17:1a:44:cc:b6:cf:40:28:
                    51:f1:c4:6b:9d:4a:57:f9:56:05:1a:3f:ba:59:b4:
                    ... (dalsza część nie została pokazana)
```

If the signature is not verified as correct (modifications have been made to the content of the CSR file) or the remaining data does not correspond to the scheme required by KDPW, the CSR will be rejected.

Verifying key compatibility

The compatibility of the files generated in the process of obtaining a certificate can be verified by checking the compatibility of the public key extracted from each of the files. If the public key matches for all the files, this means that they are compatible and relate to a single certificate. This can also help to determine whether the private key corresponds to the certificate, in case there is uncertainty about their relationship.

To simplify the private key comparison process, the SHA-256 hash function can be used for the extracted values.

To calculate the hash function for individual files using OpenSSL, use the following commands:

- SHA-256 of the public key based on a private key

```
openssl pkey -pubout -in .\private.key | openssl sha256
```

- SHA-256 of the public key based on a Certificate Signing Requets (CSR)

```
openssl req -pubkey -in .\request.csr -noout | openssl sha256
```

- SHA-256 of the public key based on a certificate

```
openssl x509 -pubkey -in .\certificate.pem -noout | openssl sha256
```

The functions should be called independently.
The result of each function is the hash value calculated for the public key extracted from each file. The result takes the following form.

```
SHA2-256(stdin)=
afd5b3b3739493c373024416a60d42676227007a6c62dcdfa72a96cfda3edb5c
```

If differences are found in this value, the set of files does not represent data concerning the same certificate. In such cases, it is recommended to re-generate the certificate.