Krajowy Depozyt Papierów Wartościowych S.A.
KDPW_CCP S.A.
ul. Książęca 4
00-498 Warszawa

T 22 537 93 43 | 22 537 91 27
F 22 627 31 11
E kdpw@kdpw.pl | ccp@kdpw.pl
www.kdpw.pl | www.kdpwccp.pl

SCP/ZW/1/2024
CCP/ZW/132/2024

Warszawa, 7th February, 2024

To: KDPW Participants
KDPW_CCP Participants

Re.: Modifications to IT systems in the area of A2A communication

Dear Madam or Sir,

With reference to the information communicated in letters of 3 October 2023 (our ref.: SCP/ZW/3/2023 and CCP/ZW/675/2023) and 20 March 2023 (our ref.: SCP/ZW/1/2023 and CCP/ZW/242/2023), announcing the KDPW Group's planned modifications to IT systems, below please find a description of the scope of the planned modifications in the area of the use of electronic certificates in A2A communication.

The first stage of the project, implemented in 2023, covered modifications to the areas of A2A communication in the EMIR Trade Repository, SFTR Trade Repository, and ARM services (with the exception of communication carried out within the SWI). As part of the modifications, an application was developed to manage the electronic certificates used to authenticate systems for MQ-based communications, the encryption algorithms were changed, and the scheme used for their generation was standardised. Uniform rules were introduced for the segregation of services within A2A communication, uniform naming in queue configuration areas, and the management of access to the test environments.
Under the adopted model, an electronic certificate is used to establish a secure encrypted system connection between a participant and the KDPW Group's IT systems. A certificate is issued for a participant's institution code, which is a four-character identifier of the participant in a service or services, according to an established scheme. The handling of certificates, in particular the

submission of certificate requests, the downloading of certificates, and the cancellation of certificates, takes place in the A2A Certificates application, available on the Services Portal (https://online.kdpw.pl) which, with the implementation of the second stage of the project, will also be available to KDPW direct participants. The application will also be made available to KDPW_CCP clearing members. The A2A Certificates application can only be accessed by a person authorised by the participant to handle certificates on its behalf. The authentication and the authorisation to the application follow the rules applicable to all applications on the Service Portal. Each person authorised by the participant may request certificates active for a given code to be generated. It is possible to re-download a generated certificate and to revoke a selected certificate.

The work currently underway, in the second stage of the project to modernise and standardise A2A communication, includes the implementation of modifications to SWI communication (carried out in accordance with the current SWI Rules), i.e., the area concerning services for KDPW direct participants (including ARM services and the Compensation Scheme) and KDPW_CCP clearing members. In practice, this involves the introduction of separate MQ queues for individual services: the depository and settlement system, the Compensation Scheme, the ARM (using the ESDK protocol), and the clearing services provided by KDPW_CCP. SWI communication carried out in the ARM service without the use of the ESDK protocol will be switched to a dedicated access point for this type of connection, launched during the first stage of the project.

At the KDPW Group level, authentication to MQ communication will be separated. Consequently, separate certificates will have to be requested in order to establish A2A communication with KDPW and with KDPW_CCP.

In addition to the change of connection parameters and the modification of the way electronic certificates are issued, it is planned as part of the modifications to discontinue the requirement to sign ESDK messages with an issued certificate. Consequently, the ESDK protocol frame will be modified and the process of ensuring the integrity of transmission will be based on in-built MQ queue support mechanisms.

A detailed description of the modifications and the documentation of the solutions is provided in an attachment to this letter.

The roll-out of the complete solution is scheduled for December 2024. With the roll-out, KDPW direct participants and KDPW_CCP clearing members will get access to the applications supporting the issuance of certificates while the issuance of certificates in the legacy authorisation centre will be discontinued. All certificates previously downloaded can be used until the communication is switched to the new upgraded solution or the certificates are revoked.

The participants will be switched to the new A2A communication model after the roll-out of the modifications, starting in December 2024, within defined switchover windows (the switchover window plan will be communicated at the final stage of the project), separately for the test environments (TST and EDU) and for the production environment (PRD). In each switchover window,

communication for a group of institution codes will be switched according to the participants' declarations. In order to minimise the impact of the switchover on business processes, the work in the production environment will be scheduled for Saturdays only.

Based on the experience of the first stage, we anticipate that the switchover of all institution codes should be completed within 3-4 months. Once the process is complete, all certificates issued in the legacy authorisation centre will be revoked.

The roll-out of these functionalities will entail formal modifications in the KDPW and KDPW_CCP regulations.

We will provide details of the process of switching communication to the new model and the planned regulatory changes at a further stage of project work.

Yours sincerely,

Sławomir Panasiuk
Vice President of the Management Board

Enclosures:
1/ Configuration specification for MQ A2A connections
2/ Description of the ESDK protocol used for A2A communication
3/ Instructions for downloading the A2A certificates used for connecting to the KDPW and KDPW_CCP services
4/ Using OpenSSL to obtain a certificate for A2A communication

C/C:
Chamber of Brokerage Houses (IDM)
Council of Depositary Banks at the Polish Bank Association (RBD ZBP)
Polish Financial Supervision Authority (UKNF)

KDPW

Appendix 1: Configuration specification for MQ A2A connections

This appendix describes the parameters required to establish an MQ connection as part of A2A communication with the KDPW and KDPW_CCP services, indicating changes compared to the previous configuration and covering modifications made in the first stage of the project. Elements requiring changes compared to the previous parameters have been highlighted in bold print and underlined.

Establishing A2A connections

- Issued and downloaded certificates for A2A communication will be used to communicate with all services where A2A communication is foreseen and where the entity is acting under the institution code assigned to the certificate. The certificate will enable the establishment of encrypted TLS communication and authentication within the MQ communication channels dedicated to the institution code.
- At A2A communication level, communication will be separated by service supported by the KDPW Group's dedicated IT solutions. This means that dedicated MQ queues will be created for each service (separately for each direction of information exchange).
- The separation of communication under an institution code will involve separate queues for A2A communication for the following services where A2A communication is available (queues will be configured only for entities actually acting and using A2A communication in the service):
    - EMIR – EMIR Trade Repository reporting services (until REFIT deployment)
    - ETR – EMIR Trade Repository reporting services (after REFIT deployment)
    - ARM – ARM reporting services
    - SFTR – SFTR Trade Repository reporting services
    - LEI – LEI issuance automation services
    - CSD – services provided within the depository and settlement system
    - ICS – Compensation System services
- The separation of communication within the services of the KDPW Group companies, particularly the separation of KDPW_CCP services based on a separate electronic certificate and connection parameters, including separate queues for A2A communication for the KDPW_CCP services:
    - CCP – KDPW_CCP clearing services

Technical connection parameters

- The following minimum requirements are expected as the security standards for telecommunications connections using VPN/IPSec technology:
    - Communication protocol – **IKEv2/IPSec (ESP)**
    - Integrity (Hashing Algorithm) – SHA-256
    - Encryption Algorithm – AES-256
    - Key exchange – Diffie-Hellman Group 19
- Parameters of MPLS network (unchanged from the previous settings):
    - Network type – L3
    - Routing – BGPv4

KDPW

MQ connection and communication parameters used for connection based on ESDK protocol

- MQ manager name for A2A communication with KDPW services:
  - Name for PRD and BCM environments – **A2AEPRD**
  - Name for EDU and TST environments – **A2AETST**
- MQ manager name for A2A communication with KDPW_CCP services:
  - Name for PRD and BCM environments – **CCPA2AEPRD**
  - Name for EDU and TST environments – **CCPA2AETST**
- TCP/IP addressing **will differ from currently used** - IP addressing and port numbers will be indicated later.
- MQ manager configuration attributes different from the IBM default values:
  - CCSID – 819
  - MAXMSGL – 104 857 600
  - VERSION – **09030015**
- MQ channel name schema - ***prefix.senv.code.con***:
  - prefix – MQ channel name prefix:
    - A2AE – KDPW services with ESDK protocol
    - CCPA2AE – KDPW_CCP services with ESDK protocol
  - senv – environment name (PRD, EDU, TST, BCM)
  - code – participant code
  - con – connection type:
    - C – server connection (*SVRCN) for client-to-server,
    - KP – KDPW-Participant for server-to-server, receiver (*RCVR) on Participants' end,
    - PK – Participant-KDPW for server-to-server, sender (*SDR) on Participants' end
- MQ channel configuration attributes different from the IBM default values:
  - COMPMSG – ZLIBFAST
  - DISCINT – 6000
  - MAXMSGL – 104 857 600
  - SSLCIPH – **TLS_AES_256_GCM_SHA384**
  - SSLPEER – Common Name (CN) of the certificate of the other party in the connection
    - PRD environment – CN=**A2AEPRD** (for KDPW_CCP services CN=**CCPA2AEPRD)**
    - EDU environment – CN=**A2AETST** (for KDPW_CCP services CN=**CCPA2AETST)**
    - TST environment – CN=**A2AETST** (for KDPW_CCP services CN=**CCPA2AETST)**
    - BCM environment – CN=**A2AEPRD** (for KDPW_CCP services CN=**CCPA2AEPRD)**
- MQ configuration attributes different from the IBM default values (unchanged from the previous settings):
  - DEFPSIST – YES
  - MAXMSGL – 104 857 600
- MQ queue name schema - ***srv.env.code.direction***
  - srv – service (available services: EMIR, ETR, ARM, SFTR, LEI, ICS, CSD, CCP)
  - senv – environment name (PRD, EDU, TST, BCM)
  - code – participant code

  - direction – MQ direction:
    - KP – messages from KDPW to the Participant
    - PK – messages from the Participant to KDPW
- MQ name schema for additional services - ***srv.env.code.direction.postfix***
  - srv – service code
  - senv – environment name
  - code – participant code
  - direction – MQ direction (KP, PK)
  - postfix – function code according to the service regulation
- Message encoding parameters for client applications (unchanged from the previous settings):
  - CodedCharSetId = 1208

KDPW

Appendix 2: Description of the ESDK protocol used for A2A communication

This appendix represents the documentation of the ESDK protocol used in some of the KDPW Group's services for A2A communication using MQ queues.

In order to accommodate the modifications to electronic certificates (in particular the cryptographic algorithms and their use in the authentication process) and the standardisation of the A2A communication rules, the use of an electronic signature in the messages will no longer be required in the protocol. This change consist merely in the removal of the field dedicated to the signature from the frame and, consequently, discontinuing the checks of the signature. The integrity of transmission will be based on the in-built MQ queue support mechanisms.
The other elements of the protocol remain unchanged.

Key information

Electronic communication with KDPW via the A2A communication interface is based on MQ connections set up with a dedicated MQ queue manager. A2A communication is based on consistent and uniform connection setup rules, regardless of the protocols used in the different services. The A2A communication model requires separation of message exchange in the different business services, which means that separate MQ queues identified by their name are provided for each service. As the A2A communication rules allow for a specific protocol or protocols to be established at the level of individual services for the exchange of information with participants, the separation also applies to the handling of protocols within each service. A communication protocol in this case is to be understood as a set of strict rules and data formats, which are required to successfully set up communication on the basis of a standard MQ connection.
One of the protocols offered as part of A2A communication is the ESDK protocol, dedicated to communication with direct participants of KDPW and KDPW_CCP. The protocol is based on a dedicated frame, ensuring technical support for checks of the communication and confirmation of the receipt of the message at the pre-processing stage. The protocol allows the transmission of any content based on standardised message type designations and communication party identifiers.

ESDK protocol description

ESDK communication protocol defines the following parameters :

- ESDK message format,
- ESDK message types,
- ESDK processing procedures for individual message types.

ESDK message format

Field name	Length	Type
Message number	9	N
Date	10	A
Time	8	A
Recipient's ID	10	A
Sender's ID	10	A
Message type	24	A
Message subtype	4	A
Reserved area	20	A
Data length	8	N
Data	Data length	B

Types:

A – character field
B - binary field
N – numeric field

Each message is unambiguously identified by the fields:

- Message number,
- Date,
- Sender's ID.

Message number: successive number of the sender's message identified by Sender's ID. The successive number is unique (for any given sender) within a single day,

Date: date of generating the message in the format YYYY-MM-DD,

Time: time of generating the message in the format HH:MM:SS,

Recipient's ID: recipient's identifier, in the format SDK.TTTTNN,

where:
- **TTTT** - participant code,
- **NN** - successive number of the identifier for a given participant.

Sender's ID: sender's identifier, in the format SDK.TTTTNN,

where:
- **TTTT** - participant code,
- **NN** - successive number of the identifier for a given participant.

Message type: defines the message type (padded with spaces to the right),

Message subtype: defines the message subtype. Default value for this field is '0000'. In content messages, the first character in this field may be attributed the following values:

- 'T' - for messages sent in a fixed-field format,
- 'X' - for messages sent in XML format,

- '0' - for non-defined message format

Sender's ID: an area which may in the future be filled with additional header data,
Data length: length of the **Data** field**,**
Data: data transferred in the form of a message.

ESDK message types

The **Message type** field may take the following values:

- **esdk.acc.001.01** confirmation of message acceptance,
- **esdk.rjc.001.01** information on message rejection,
- **esdk.tst.001.01** test message,
- message type generated by the KDPW Group's services and/or by the participant.

Messages whose 4 initial characters are assigned the "esdk" value are hereinafter referred to as technical messages. The Data field in technical messages (excluding esdk.tst.001.01 message) has a specific format.

Structure of the esdk.acc.001.01 message

Structure of the **Data** field in **esdk.acc.001.01** messages**:**

Field name	Length	Type
Message number	9	N
Data	10	A
Sender's ID	10	A
Acceptance date	10	A
Rejection time	8	A

The above structure identifies messages accepted in the ESDK system and informs about the date and time of message acceptance.

Structure of the esdk.rjc.001.01 message

Structure of the **Data** field in **esdk.rjc.001.01** messages**:**

Field name	Length	Type
Message number	9	N
Data	10	A
Sender's ID	10	A
Acceptance date	10	A
Rejection time	8	A

Error code	10	A
Description of error	256	A

The ***Message number***, ***Date***, ***Sender's ID*** fields identify rejected message.
The ***Rejection date*** and ***Rejection time*** fields inform about the date and time of message rejection.
The ***Error code*** and ***Error description*** fields describe the reason for message rejection.

Processing different message types

All messages received by the ESDK system are checked as follows:

- checking the message structure against the required format,
- checking that the message identifier is unique,
- checks specific to the type of message.

As a result of the checks, the message is either accepted or rejected. Information on all messages received and sent is recorded in the message log.
KDPW (KDPW_CCP) performs the following checks in the handling of ESDK protocol technical messages:

Messages esdk.acc.001.01:

- if message **esdk.acc.001.01** is accepted, no further action is taken,
- if message **esdk.acc.001.01** is rejected, the system sends information about the message and the result of its checks to the system administrator.

Messages esdk.rjc.001.01:

Upon receipt of message esdk.rjc.001.01, the system sends information about the message and the result of its checks to the system administrator. Depending on the content of the message, the administrators take specific clarification actions.

Messages esdk.tst.001.01:

- if message **esdk.tst.001.01** is accepted, message **esdk.acc.001.01** is sent to the sender,
- if message **esdk.tst.001.01** is rejected, message **esdk.rjc.001.01** is sent to the sender with the rejection and the reason.

Content messages:

- if a content message is accepted as compliant with the ESDK protocol, message **esdk.acc.001.01** is sent to the sender and the content message is forwarded to the dedicated service for processing. As a result of the processing, further content messages may be sent to the sender with the status and results of the processing of the message in the domain system,
- if a content message is rejected as non-compliant with the ESDK protocol, message **esdk.rjc.001.01** is sent to the sender with the rejection and the reason, and the content message is not forwarded for further processing.

KDPW

Appendix 3: Instructions for downloading the A2A certificates used for connecting to the KDPW and KDPW_CCP services

This appendix describes the A2A certificate scheme in communication with the KDPW and KDPW_CCP services and how to generate and download a certificate using dedicated applications (separately for KDPW and KDPW_CCP) available on the Services Portal (https://online.kdpw.pl).
After gaining access to this application, a person authorized by the participant can download both certificates for production communication and tests. At the same time, the application will not be available in any of the Service Portal test environments.

Electronic certificates used for A2A communication

- To establish secure communication with KDPW Group services based on MQ queues in the A2A model, KDPW and KDPW_CCP use electronic certificates based on asymmetric cryptography within the Public Key Infrastructure (PKI).
- An electronic certificate is issued for a participant's institution code and can be used to establish communication for access to all services where the participant appears under that code (separately for KDPW and KDPW_CCP). 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.
- 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
- Only certificates generated in accordance with the scheme established for A2A communications and signed by KDPW may be used to establish communications.

Description of the A2A certificate generation process

- A certificate is generated and issued using PKI mechanisms and the SHA-256 algorithm. The process of obtaining a certificate is carried out in several steps, with the security of the private key

ensured. Throughout the process, the private key of the participant requesting the certificate is not disclosed to KDPW (it should be secured within the infrastructure of the certificate owner).

- A certificate is obtained based on a Certificate Signing Request (CSR) transmitted by the applicant, generated and saved in PEM format.
- A Certificate Signing Request is generated directly by the applicant within its own infrastructure, e.g. using OpenSSL mechanisms. This requires appropriate IT expertise and access to tools for generating private keys and CSRs.
- When creating a Certificate Signing Request, the applicant must ensure that it complies with the established scheme to the extent of the key type and length and the data comprising the Distinguished Name (DN), in particular the institution code and the environment code.
- The management of electronic certificates to the extent of requesting a certificate, downloading a certificate and revoking a certificate, is possible through a dedicated certificate management application available within the Services Portal (https://online.kdpw.pl). Only a person authorised by the participant for certificate management on its behalf can access the application.
- A Certificate Signing Request is transmitted in the form of a request in the dedicated application, to which it is attached in text form. In the preparation of the request, the application verifies the conformity of the information contained in the CSR with the certificate scheme, including the institution code in the context of which it is submitted.
- A Certificate Signing Request must be created in PKCS#10 format using PEM encoding.
- Once the CSR has been processed, the application will enable the download of a certificate generated and signed by KDPW, which can be used within the participant's infrastructure to establish a TLS connection and authenticate in the MQ communication channel within the A2A communication. The applicant may, within its own infrastructure, combine the received certificate with a previously generated private key in the form of a PKCS#12 container.

NOTE: Continuous access to A2A channel services requires a valid certificate. Participants should monitor the validity of certificates and request the generation of a new certificate in due time. Participants may also request more than one certificate for a given institution code.

Participants should keep electronic certificates and private keys in a secure location. In the event of loss of the private key or in the event of any security breach, participants must revoke the certificate using the dedicated application. Once revoked, the certificate will be placed on the Certificate Revocation List (CRL), which is published at: http://pki.kdpw.pl/crl/kdpw-cck1.crl.

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.