Federal Office
for Information Security

Technical Guideline TR-03129-4

Protocols for the Management of Certificates and CRLs in Public-Key-Infrastructures (PKIs)

Part 4: Smart Metering Infrastructure

Version 1.3.1

Federal Office for Information Security
Post Box 20 03 63
D-53133 Bonn

E-Mail: eid@bsi.bund.de
Internet: https://www.bsi.bund.de
© Federal Office for Information Security 2017

Table of Contents

1 Introduction 4
1.1 Terminology 4
1.2 Authentication, Confidentiality and Integrity Protection 4
2 Protocols for Smart Meter Certificate Management 5
2.1 Messages 5
2.2 Parameters and Return Codes 5
2.2.1 Parameters 5
2.2.2 Return Codes 6
2.3 Distribution of Certificates 8
2.3.1 GeneralMessage 8
2.3.2 RequestCertificate 9
2.3.3 GetCertificateChain 10
2.3.4 SendCertificates 10
2.3.5 CertificateRevocationRequest 11
2.3.6 GetServiceStatus 12
2.3.7 UpdateDeviceAdmin 12
2.4 Implementation as Web Services 13
2.4.1 Services Implemented by the Root-CA 13
2.4.2 Services Implemented by the Sub-CAs 13
2.4.3 Services Implemented by End-Users 14
A WSDL and XML Schema specifications 15
Reference Documentation 16
Keywords and Abbreviations 17

Figures

Tables

Table 1: Key words 4

1 Introduction

The Smart Meter Public Key Infrastructure forms a three-tier hierarchy with an official trust anchor (the Root-CA), authorized Sub-CAs for End-User Management and Market Participants, Smart Meter Gateways or Gateway Administrators as End-Users. X.509 certificates are used to secure communication between a Smart Meter Gateway and external entities.

This technical guideline specifies PKI-related communication protocols for the distribution of X.509 certificates and related data between the Root-CA, Sub-CAs and End-Users. For specific provisions on the PKI in the context of Smart Meters, please refer to the relevant Technical Guideline [TR-03109-4].

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119]. The key word “CONDITIONAL” is to be interpreted as follows:

CONDITIONAL: The usage of an item is dependent on the usage of other items. It is therefore further qualified under which conditions the item is REQUIRED or RECOMMENDED.

When used in tables (profiles), the key words are abbreviated as shown in Table 1.

Key word		**Abbrev.**
MUST / SHALL	REQUIRED	m
MUST NOT / SHALL NOT	–	x
SHOULD	RECOMMENDED	r
MAY	OPTIONAL	o
–	CONDITIONAL	c

Table 1: Key words

1.2 Authentication, Confidentiality and Integrity Protection

The authentication of the sender and receiver of messages as well as the confidentiality and integrity of the contents of the messages are not considered at the level of these messages. It is assumed that authentication, integrity protection and confidentiality are guaranteed by other measures, for example by using transport layer security (TLS) [RFC 5246] with server and client authentication.

Each service provider SHALL use the underlying authentication mechanism to verify whether (and to what extent) the caller is authorised to use the service and deny access otherwise. Security relevant information MUST always be protected by appropriate encryption and integrity protection mechanisms.

2 Protocols for Smart Meter Certificate Management

The implementation of communication protocols shall be realized as web services using SOAP-messages, which must be in conformance with the attached WSDL files.

Some of the specified protocols allow messages to be processed either synchronously or asynchronously. If a message is processed synchronously, the result of processing the message (if any) is given back to the sender contained in the output parameters of the message while terminating message processing. If the message is processed asynchronously, only a return code is given back to the sender while terminating message processing. In this case the result of processing the message (if any) is sent by the receiver (of the original message) to the sender (of the original message) contained in input parameters of a corresponding callback message.

If a message is processed asynchronously and the result is sent back using a callback message, both participating entities must work as a client in order to send a message as well as a server in order to receive a message. A sender of a message may indicate in a message that it does not accept to receive a corresponding callback message containing the result of the original message. In this case the receiver of the message may process the message synchronously, i.e. the output parameters of the message must contain the final results of processing the message. Polling mechanisms are not available in this protocol, i.e. in the case of an asynchronous message processing the sender cannot ask the receiver frequently if the results of message processing are available for download using messages of this protocol.

2.1 Messages

In the following abstract definitions of the messages used for the protocols will be given. For each message, the following characteristics are described:

- Intended use
- Input parameters
- Output parameters

  → including possible return codes

Unless specified differently, each message can be processed by the receiving entity either asynchronously or synchronously. If a message is processed asynchronously, the output parameters of the message are not used. In this case, the result of processing the message will be sent to the originator using another (response) message.

2.2 Parameters and Return Codes

2.2.1 Parameters

The following parameters are frequently used as input or output parameters:

- `callbackIndicator`

  With this parameter the originator of the message informs the receiver if it can handle callbacks as response to this message. If the originator can handle callbacks, this parameter MUST be set to `callback_possible`. In this case, the receiver can decide if it processes this message synchronously or asynchronously. If the receiver processes this message asynchronously, it will send the response using the appropriate callback message. If the originator cannot handle callbacks, this parameter MUST be set to `callback_not_possible`.

- `messageID`

This parameter contains the identification of the message. It MUST identify the message uniquely within all messages of the originator. If a response message will be sent to the originator as a result of this message, the response message SHALL contain the same `messageID`. Hence an incoming response message can be assigned to the correct original message. Construction and allocation of the `messageID` can be decided by the originator. This parameter is REQUIRED if the originator of the message indicates, that it can handle a callback as response to this message (i.e. parameter `callbackIndicator = callback_possible`). It MUST be omitted, if the originator of the message indicates, that it cannot handle callbacks as response to this message (i.e. parameter `callbackIndicator = callback_not_possible`).

- `returnCode`

  This parameter contains a status code about the result of processing the current message.

- `returnCodeMessage`

  This parameter may include additional information, which are not covered by the already defined return codes. The maximum length of a message MUST NOT exceed 1024 bytes.

- `statusInfo`

  This parameter is only used in an asynchronous response message and contains a status code about the result of processing the corresponding request message. In this context, it assumes the function of a return code of a synchronous message.

- `statusInfoMessage`
- This parameter may include additional information, which are not covered by the already defined `statusInfo` codes. The maximum length of a message MUST NOT exceed 1024 bytes.
- `certReq`

  This parameter contains the certificate request. Its construction and coding MUST follow the specification in appendix C of [TR-03109-4].

- `certificateSeq`

  This parameter contains one or more certificates, if the message has been processed successfully. It is REQUIRED if certificates have to be sent with the response. It MUST be missing if no certificates will be sent with the message. If the parameter contains a chain of certificate they should be ordered starting with the hierarchically highest certificate.

- `certRevReq`

  This parameter contains the certificate revocation request. Its construction and coding MUST follow the specification in appendix C of [TR-03109-4].

- `updDevAdminReq`

  This parameter contains the data structure necessary to update the gateway administrator for one or more SMGW devices. Its construction and coding MUST follow the specification in appendix C of [TR-03109-4].

2.2.2 Return Codes

The following return codes are used frequently in this section:

- `ok_syntax`

  The reception of the SOAP-message is acknowledged. The syntax of the message has been verified successfully. The further processing of the message will be done asynchronously. The result of the processing will be sent to the web service URL determined from the TLS-certificate of the sender of the request-message. This URL must be provided on the client side and should have been exchanged between contracting parties within the initial registration process (see also [TR-03109-4]). The TLS-certificate of the request-message sender must be used to identify that sender and to look up the web service URL to be called.

- `ok_received_correctly`

  The message has been received and processed synchronously. No output is generated.

- `ok_cert_available`

  The message has been processed successfully. The parameter `certificateSeq` contains one or more certificates.

- `failure_syntax`

  The received SOAP-message is syntactically not correct.

- `failure_incorrect_request`

  The received request-message was incorrectly formatted and could not be processed by the application.

- `failure_synchronous_processing_not_possible`

  The sender has indicated that he does not accept callback messages, hence the message must be processed synchronously by the receiver. But the receiver cannot process this message synchronously. In this case further procedures must be determined by organisational measures.

- `failure_messageID_unknown`

  The contained `messageID` cannot be matched with a message formerly sent.

- `failure_other_error`

  An error occurred which is not covered by an already defined `returnCode` or `returnCodeMessage` (or `statusInfo` or `statusInfoCode`). If `failure_other_error` is thrown, a message SHOULD be sent to the recipient of the return code by using the command `GeneralMessage`, which includes a meaningful description of the error.

- `failure_internal_error`

  This code should only be returned, if an unexpected error occurred, e.g. in case the application hangs or crashes.

- `failure_inner_signature`

  The verification of the inner signature of the actual certificate request failed. `returnCodeMessage` should contain additional information why the verification failed (e.g. if the signature was expired or if the signature was incorrectly encoded).

- `failure_outer_signature`

  The verification of the outer signature of the actual certificate request failed. `returnCodeMessage` should contain additional information why the verification failed (e.g. if the signature was expired or if the signature was incorrectly encoded).

- `failure_authorization_signature`

  The verification of the authorization signature of the actual certificate request failed. `returnCodeMessage` should contain additional information why the verification failed (e.g. if the signature was expired or if the signature was incorrectly encoded).

- `failure_request_not_accepted`

The message has been processed correctly but the request has not been accepted (i.e. the applicant has been suspended by the CA). In this case the applicant must manually contact the (Sub-)CA to resolve the issue.

- `failure_unauthorized_request`

  The type of the requested certificate does not match the client requesting the certificate. E.g. a GWA (gateway administrator) requested a certificate of type SMGW-G (Smart-Meter-Gateway approval certificate)

- `failure_domain_parameters`

  The domain parameters contained in the request are not allowed by the policy of the PKI, e.g. a wrong signature algorithm or a wrong named curve has been chosen.

- `failure_cert_not_available`

  The corresponding message has been processed correctly but the requested certificates are not available.

2.3 Distribution of Certificates

The following messages are used for the management of X.509 certificates within the Smart Meter PKI.

2.3.1 GeneralMessage

This message is sent by an entity in the PKI hierarchy to another entity in order to send notifications or other general human readable text message.

Input parameters:

- `callerID` **(REQUIRED)**

  This parameter contains the identifier of the originating entity. The value SHALL be unique within the PKI hierarchy.

- `messageID` **(REQUIRED)**
- `subject` **(REQUIRED)**

  This parameter contains the subject of the message. The subject SHOULD briefly describe the content of the message body.

- `body` **(REQUIRED)**

  This parameter contains the body of the message. The body SHALL be human readable plain text which is not intended for direct automated processing.

Output parameters:

- `returnCode` **(REQUIRED)**

  The following return codes are possible:

  - `ok_received_correctly`
  - `failure_syntax`
  - `failure_other_error`
  - `failure_internal_error`

- `returnCodeMessage` **(OPTIONAL)**

2.3.2 RequestCertificate

This message is used by an End-User or by a Sub-CA for requesting the generation of a new certificate for one of its keys from a Sub-CA or the Root-CA, respectively. If the parent (Sub-)CA processes this message asynchronously, it MUST must respond using the `SendCertificates` message.

Input parameters:

- `callbackIndicator` **(REQUIRED)**
- `messageID` **(CONDITIONAL)**
- `certReq` **(REQUIRED)**

Output parameters:

- `certificateSeq` **(CONDITIONAL)**
- `returnCode` **(REQUIRED)**

  The following return codes are possible:

  - `ok_syntax`
  - `ok_cert_available`
  - `failure_syntax`
  - `failure_incorrect_request`
  - `failure_synchronous_processing_not_possible`
  - `failure_request_not_accepted`
  - `failure_unauthorized_request`
  - `failure_other_error`
  - `failure_internal_error`
  - `failure_inner_signature`
  - `failure_outer_signature`
  - `failure_authorization_signature`
  - `failure_domain_parameters`
- `returnCodeMessage` **(OPTIONAL)**

Remarks:

In case of end-user certificates, it is recommended to supply the type of the certificate request in the SOAP-header of the message by using the parameter:

- `certType`

  Depending on the certificate request, the following values should be used:

  - for SMGW operational certificates: `SMGW-W`
  - for SMGW seal of approval certificates (Gütesiegelzertifikate): `SMGW-G`
  - for GWA certificates: `GWA`
  - for GWH certificates: `GWH`

  - for EMT certificates: `EMT`
  - for Sub-CA certificates: `Sub-CA`

2.3.3 GetCertificateChain

This message is sent by an entity who is trying to check a certificate to the (Sub)-CA where the certificate has been issued in order to retrieve all relevant X.509 certificates required for the verification. The Sub-(CA) should return a chain of certificates including all valid link certificates, beginning with the first Root-CA certificate, and ending with the indicated certificate of the (Sub-)CA.

If the parent (Sub-)CA processes this message asynchronously, it MUST respond using the `SendCertificates` message.

Input parameters:

- `callbackIndicator` **(REQUIRED)**
- `messageID` **(CONDITIONAL)**
- `certReference` **(REQUIRED)**

  The certReference MUST contain the distinguished name of the issuer from the certificate that needs to be verified.

Output parameters:

- `certificateSeq` **(CONDITIONAL)**
- `returnCode` **(REQUIRED)**

  The following return codes are possible:

  - `ok_syntax`
  - `ok_cert_available`
  - `failure_syntax`
  - `failure_synchronous_processing_not_possible`
  - `failure_other_error`
  - `failure_internal_error`
  - `failure_cert_not_available`
- `returnCodeMessage` **(OPTIONAL)**

Remarks:

If the message is processed successfully and accepted the receiver MUST send all relevant CA certificates within the response, either in the output parameter `certificateSeq` (synchronous processing) or in the corresponding response message `SendCertificates` (asynchronous processing).

2.3.4 SendCertificates

If a certification authority processes one of the message `RequestCertificate or GetCertificateChain` asynchronously, it uses a response message `SendCertificates` to communicate the result of its processing. It always sends the response message to the WebService URL of the applicant which is identified by their TLS certificate.

This message can also be used to notify registered entities about the availability of new X.509 certificates. In this case the `messageID` must be omitted.

This message itself MUST always be processed synchronously by its receiver.

Input parameters:

- `messageID` **(CONDITIONAL)**
- `statusInfo` **(REQUIRED)**

  The following status codes are possible:

  - `ok_cert_available`
  - `failure_syntax`
  - `failure_incorrect_request`
  - `failure_request_not_accepted`
  - `failure_unauthorized_request`
  - `failure_other_error`
  - `failure_internal_error`
  - `failure_inner_signature`
  - `failure_outer_signature`
  - `failure_authorization_signature`
  - `failure_domain_parameters`
  - `failure_cert_not_available`
- `statusInfoMessage` **(OPTIONAL)**
- `certificateSeq` **(CONDITIONAL)**

Output parameters:

- `returnCode` **(REQUIRED)**

  The following return codes are possible:

  - `ok_received_correctly`
  - `failure_syntax`
  - `failure_messageID_unknown`
  - `failure_other_error`
  - `failure_internal_error`
- `returnCodeMessage` **(OPTIONAL)**

2.3.5 CertificateRevocationRequest

In order to revoke or suspend a SMGW certificate, this message must be sent by an entity which is responsible for the operation of the SMGW to the certificate issuing Sub-CA of a SMGW. Each revocation request MUST include a revocation reason by supplying the corresponding reason code.

A Sub-CA receiving this message MUST either process it synchronously or reject it.

Input parameters:

- `callbackIndicator` **(REQUIRED)**

  must be set to `callback_not_possible`
- `messageID` **(CONDITIONAL)**
- `certRevReq` **(REQUIRED)**

Output parameters:

- `returnCode` **(REQUIRED)**

  The following return codes are possible:
  - `ok_received_correctly`
  - `failure_syntax`
  - `failure_incorrect_request`
  - `failure_request_not_accepted`
  - `failure_other_error`
  - `failure_internal_error`
- `returnCodeMessage` **(OPTIONAL)**

2.3.6 GetServiceStatus

This message is sent by a requesting entity in the PKI hierarchy to a receiving entity in order to test the web service communication between them and to retrieve the current status of the receiving entity.

If the receiving entity's application (e.g. the CA) is operational it SHALL answer with return code `ok_received_correctly`. If the SOAP-message has been received properly, but the application behind the web service is not operational, the receiving entity SHALL answer with return code `failure_other_error` and return the actual application status within `returnCodeMessage`.

Input parameters:

- `messageID` **(REQUIRED)**

Output parameters:

- `returnCode` **(REQUIRED)**

  The following return codes are possible:
  - `ok_received_correctly`
  - `failure_syntax`
  - `failure_other_error`
  - `failure_internal_error`
- `returnCodeMessage` **(OPTIONAL)**

2.3.7 UpdateDeviceAdmin

In order to update the gateway administrator (GWA) for certain SMGW devices, this message must be sent by an entity which is currently in charge of the SMGW towards the certificate-issuing Sub-CA. A single update package MUST contain at least one, but MAY contain multiple Common Names of SMGW certificates to update the GWA in charge for.

A Sub-CA receiving this message MUST either process it synchronously or reject it.

If the message is processed successfully by a Sub-CA, the newly assigned GWA in charge becomes the only valid entity allowed to revoke, suspend and unsuspend certificates of the SMGW.

Input parameters:

- `callbackIndicator` **(REQUIRED)**

  must be set to `callback_not_possible`
- `messageID` **(CONDITIONAL)**
- `updDevAdminReq` **(REQUIRED)**

Output parameters:

- `returnCode` **(REQUIRED)**

  The following return codes are possible:
  - `ok_received_correctly`
  - `failure_syntax`
  - `failure_incorrect_request`
  - `failure_request_not_accepted`
  - `failure_other_error`
  - `failure_internal_error`
- `returnCodeMessage` **(OPTIONAL)**

2.4 Implementation as Web Services

It is RECOMMENDED that the protocol given in this chapter is realised as a Web Service using the SOAP messages described in the attached WSDL specifications.

2.4.1 Services Implemented by the Root-CA

A Root-CA MUST implement a Web Service supporting the following messages:

- `RequestCertificate`
- `GetCertificateChain`
- `GeneralMessage`
- `GetServiceStatus`

2.4.2 Services Implemented by the Sub-CAs

For the communication with its associated End-Users a Sub-CA MUST implement a Web Service supporting the following messages:

- `RequestCertificate`
- `GetCertificateChain`

- `CertificateRevocationRequest`
- `UpdateDeviceAdmin`

If a Sub-CA supports callback messages sent by the Root-CA, the Web Service of the Sub-CA MUST in addition also support the following (callback) message:

- `SendCertificates`

For communication with both the Root-CA and its associated End-Users, a Sub-CA MUST implement the following message:

- `GeneralMessage`
- `GetServiceStatus`

2.4.3 Services Implemented by End-Users

An End-User has to implement the corresponding Web Service only if it supports callback messages sent by their Sub-CA. In this case it MUST implement a Web Service supporting the following (callback) messages:

- `SendCertificates`
- `GeneralMessage`
- `GetServiceStatus`

A WSDL and XML Schema specifications

The corresponding XML schema definitions and WSDL descriptions are part of this specification and provided as separate files.

Reference Documentation

TR-03109-4	BSI TR-03109-4: Technische Richtlinie BSI TR-03109-4: Smart Metering PKI - Public Key Infrastruktur für Smart Meter Gateways, Version 1.2, 2016
RFC 2119	RFC 2119: Key words for use in RFCs to indicate requirement levels, 1997
RFC 5246	RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2, 2008

Keywords and Abbreviations

Abbreviations........ 17