Stream: Internet Engineering Task Force (IETF)
RFC: 9483
Category: Standards Track
Published: November 2023
ISSN: 2070-1721
Authors: H. Brockhaus, *Siemens*; D. von Oheimb, *Siemens*; S. Fries, *Siemens*

RFC 9483
Lightweight Certificate Management Protocol (CMP) Profile

Abstract

This document aims at simple, interoperable, and automated PKI management operations covering typical use cases of industrial and Internet of Things (IoT) scenarios. This is achieved by profiling the Certificate Management Protocol (CMP), the related Certificate Request Message Format (CRMF), and transfer based on HTTP or Constrained Application Protocol (CoAP) in a succinct but sufficiently detailed and self-contained way. To make secure certificate management for simple scenarios and constrained devices as lightweight as possible, only the most crucial types of operations and options are specified as mandatory. More specialized or complex use cases are supported with optional features.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9483.

Copyright Notice

Table of Contents

1. Introduction

This document specifies PKI management operations supporting machine-to-machine and IoT use cases. Its focus is to maximize automation and interoperability between all involved PKI entities, ranging from end entities (EEs) over any number of intermediate PKI management

entities, such as registration authorities (RAs), to the Certificate Management Protocol (CMP) [RFC4210] endpoints of certification authority (CA) systems. This profile makes use of the concepts and syntax specified in CMP [RFC4210] [RFC9480] [RFC9481], Certificate Request Message Format (CRMF) [RFC4211] [RFC9045], Cryptographic Message Syntax (CMS) [RFC5652] [RFC8933], HTTP transfer for CMP [RFC6712], and CoAP transfer for CMP [RFC9482]. CMP, CRMF, and CMS are feature-rich specifications, but most application scenarios use only a limited subset of the same specified functionality. Additionally, the standards are not always precise enough on how to interpret and implement the described concepts. Therefore, this document aims to tailor the available options and specify how to use them in adequate detail to make the implementation of interoperable automated certificate management as straightforward and lightweight as possible.

While this document was being developed, documents intended to obsolete RFC 4210 [PKIX-CMP] and RFC 6712 [HTTP-CMP] were posted, and they include the full set of changes described in CMP Updates [RFC9480].

1.1. How to Read This Document

This document has become longer than the authors would have liked it to be. Yet apart from studying Section 3, which contains general requirements, the reader does not have to work through the whole document. The guidance in Sections 1.9 and 7 should be used to figure out which parts of Sections 4 to 6 are relevant for the target certificate management solution, depending on the PKI management operations, their variants, and types of message transfer needed.

Since conformity to this document can be achieved by implementing only the functionality declared mandatory in Section 7, the profile can still be called lightweight because, in particular for end entities, the mandatory-to-implement set of features is rather limited.

1.2. Conventions and Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The term "PROHIBITED" is to be interpreted to mean that the respective ASN.1 field **SHALL NOT** be present or used.

Technical terminology is used in conformance with [RFC4210], [RFC4211], [RFC5280], and IEEE 802.1AR [IEEE.802.1AR_2018]. The following terminology is used:

CA: Certification authority, which issues certificates.

RA: Registration authority, an optional PKI component to which a CA delegates certificate management functions, such as end entity authentication and authorization checks for incoming requests. An RA can also provide conversion between various certificate management protocols and other protocols providing some operations related to certificate management.

LRA: Local registration authority, a specific form of RA with proximity to the end entities.

Note: For ease of reading, this document also uses the term "RA" for LRAs in all cases where the difference is not relevant.

KGA: Key generation authority, an optional system component, typically colocated with an RA or CA, that offers key generation services to end entities.

EE: End entity, typically a device or service that holds a public-private key pair for which it manages a public key certificate. An identifier for the EE is given as the subject of its certificate.

The following terminology is reused from [RFC4210] as follows:

PKI management operation: All CMP messages belonging to a single transaction. The transaction is identified by the transactionID field of the message headers.

PKI management entity: A non-EE PKI entity, i.e., an RA or a CA.

PKI entity: An EE or PKI management entity.

CMP messages are referred to by the names of PKIBody choices defined in Section 5.1.2 of [RFC4210] and are further described in Section 4 of this document.

The following terms are introduced in this document:

CMP protection key: The private key used to sign a CMP message.

CMP protection certificate: The certificate related to the CMP protection key. If the keyUsage extension is present, it **MUST** include digitalSignature.

1.3. Motivation for a Lightweight Profile of CMP

CMP was standardized in 1999 and is implemented in several PKI products. In 2005, a completely reworked and enhanced version 2 of CMP [RFC4210] and CRMF [RFC4211] has been published, followed by a document specifying a transfer mechanism for CMP messages using HTTP [RFC6712] in 2012.

CMP is a capable protocol and could be used more widely. CMP [RFC4210] and CMP Updates [RFC9480] offer a very large set of features and options. On one hand, this makes CMP applicable to a very wide range of scenarios; on the other hand, a full implementation supporting all options is not realistic because this would take undue effort.

In order to reduce complexity, the set of mandatory PKI management operations and variants required by this specification has been kept lean. This limits development efforts and minimizes resource needs, which is particularly important for memory-constrained devices. To this end, when there was design flexibility to either have necessary complexity on the EE or in the PKI management entity, this profile chose to include it in the PKI management entities where typically more computational resources are available. Additional recommended PKI management operations and variants support some more complex scenarios that are considered beneficial for environments with more specific demands or boundary conditions. The optional PKI management operations support less common scenarios and requirements.

Moreover, many details of the Certificate Management Protocol have been left open or have not been specified in full preciseness. The profiles specified in Appendices D and E of [RFC4210] define some more detailed PKI management operations. Yet the specific needs of highly automated scenarios for machine-to-machine communication are not covered sufficiently.

Profiling is a way to reduce feature richness and complexity of standards to what is needed for specific use cases. 3GPP and UNISIG already use profiling of CMP as a way to cope with these challenges. To profile means to take advantage of the strengths of the given protocol while explicitly narrowing down the options it provides to those needed for the purpose(s) at hand and eliminating all identified ambiguities. In this way, the general aspects of the protocol are utilized and only the special requirements of the target scenarios need to be dealt with using distinct features the protocol offers.

Defining a profile for a new target environment takes high effort because the range of available options needs to be well understood and the selected options need to be consistent with each other and suitably cover the intended application scenario. Since most industrial PKI management use cases typically have much in common, it is worth sharing this effort, which is the aim of this document. Other standardization bodies can reference this document and further tailor the PKI management operations to their needs to avoid coming up with individual profiles from scratch.

1.4. Special Requirements of Industrial and IoT Scenarios

The profiles specified in Appendices D and E of [RFC4210] have been developed particularly for managing certificates of human end entities. With the evolution of distributed systems and client-server architectures, certificates for machines and applications on them have become widely used. This trend has strengthened even more in emerging industrial and IoT scenarios. CMP is sufficiently flexible to support them well.

Today's IT security architectures for industrial solutions typically use certificates for endpoint authentication within protocols like IPsec, TLS, or Secure Shell (SSH). Therefore, the security of these architectures highly relies upon the security and availability of the implemented certificate management operations.

Due to increasing security and availability needs in operational technology, especially when used for critical infrastructures and systems with a high number of certificates, a state-of-the-art certificate management system must be constantly available and cost-efficient, which calls for

high automation and reliability. Consequently, "Framework for Improving Critical Infrastructure Cybersecurity" [NIST.CSWP.04162018] refers to proper processes for issuance, management, verification, revocation, and audit of authorized devices, users, and processes involving identity and credential management. According to commonly accepted best practices, such PKI management operations are also required in [IEC.62443-3-3] for security level 2 and higher.

Further challenges in many industrial systems are network segmentation and asynchronous communication. Also, PKI management entities like certification authorities (CAs) are not typically deployed on-site but in a highly protected data center environment, e.g., operated according to ETSI Policy and security requirements for Trust Service Providers issuing certificates [ETSI-EN.319411-1]. Certificate management must be able to cope with such network architectures. CMP offers the required flexibility and functionality, namely authenticated self-contained messages, efficient polling, and support for asynchronous message transfer while retaining end-to-end authentication.

1.5. Existing CMP Profiles

As already stated, [RFC4210] contains profiles with mandatory and optional PKI management operations in Appendices D and E of [RFC4210]. Those profiles focus on management of human user certificates and only partly address the specific needs of certificate management automation for unattended devices or machine-to-machine application scenarios.

Both Appendices D and E of [RFC4210] focus on PKI management operations between an EE and an RA or CA. They do not address further profiling of RA-to-CA communication, which is typically needed for full backend automation. All requirements regarding algorithm support for Appendices D and E of [RFC4210] have been updated by Section 7.1 of CMP Algorithms [RFC9481].

3GPP makes use of CMP [RFC4210] in its Technical Specification 33.310 [ETSI-3GPP.33.310] for automatic management of IPsec certificates in 3G, LTE, and 5G backbone networks. Since 2010, a dedicated CMP profile for initial certificate enrollment and certificate update operations between EEs and RAs/CAs is specified in that document.

In 2015, UNISIG included a CMP profile for enrollment of TLS certificates in the Subset-137 specifying the ETRAM/ETCS online key management for train control systems [UNISIG.Subset-137].

Both standardization bodies tailor CMP [RFC4210], CRMF [RFC4211], and HTTP transfer for CMP [RFC6712] for highly automated and reliable PKI management operations for unattended devices and services.

1.6. Compatibility with Existing CMP Profiles

The profile specified in this document is compatible with Appendices D and E of [RFC4210], with the following exceptions:

- signature-based protection is the default protection; an initial PKI management operation may also use protection based on the message authentication code (MAC),

- certification of a second key pair within the same PKI management operation is not supported,
- proof-of-possession (POP) with the self-signature of the certReq containing the certTemplate (according to [RFC4211], Section 4.1, clause 3) is the recommended default POP method (deviations are possible for EEs when requesting central key generation, for RAs when using raVerified, and if the newly generated keypair is technically not capable to generate digital signatures),
- confirmation of newly enrolled certificates may be omitted, and
- all PKI management operations consist of request-response message pairs originating at the EE, i.e., announcement messages (requiring a push model, a CMP server on the EE) are excluded in favor of a lightweight implementation on the EE.

The profile specified in this document is compatible with the CMP profile for 3G, LTE, and 5G network domain security and authentication framework [ETSI-3GPP.33.310], except that:

- protection of initial PKI management operations may be MAC-based,
- the subject field is mandatory in certificate templates, and
- confirmation of newly enrolled certificates may be omitted.

The profile specified in this document is compatible with the CMP profile for online key management in rail networks as specified in [UNISIG.Subset-137], except that:

- A certificate enrollment request message consists of only one certificate request (CertReqMsg).
- [RFC4210] requires that the messageTime is Greenwich Mean Time coded as generalizedTime.

  Note: As Table 5 of [UNISIG.Subset-137] explicitly states that the messageTime is required to be "UTC time", it is not clear if this means a coding as UTCTime or generalizedTime and if time zones other than Greenwich Mean Time shall be allowed. Both time formats are described in Section 4.1.2.5 of [RFC5280].
- The same type of protection is required to be used for all messages of one PKI management operation. This means, in case the request message protection is MAC-based, the response, certConf, and pkiConf messages must also have MAC-based protection.
- Use of caPubs is not required but is typically allowed in combination with MAC-based protected PKI management operations. On the other hand, Table 12 of [UNISIG.Subset-137] requires using caPubs.

  Note: It remains unclear from UNISIG Subset-137 which certificate(s) for the caPubs field should be used. For security reasons, it cannot be used for delivering the root CA certificate needed to validate the signature-based protection of the given response message (as stated indirectly also in Section 6.3.1.5.2 b of [UNISIG.Subset-137]).
- This profile requires that the certConf message have one CertStatus element where the statusInfo field is recommended.

Note: In contrast, Table 18 of [UNISIG.Subset-137] requires that the certConf message has one CertStatus element where the statusInfo field must be absent. This precludes sending a negative certConf message in case the EE rejects the newly enrolled certificate. This results in violating the general rule that a certificate request transaction must include a certConf message (moreover, since using implicitConfirm is not allowed there either).

1.7. Use of CMP in SZTP and BRSKI Environments

In Secure Zero Touch Provisioning (SZTP) [RFC8572] and other environments using Network Configuration Protocol (NETCONF) / YANG modules, [SZTP-CSR] offers a YANG module that includes several types of certificate requests to obtain a public key certificate for a locally generated key pair. Such messages are of the form ietf-ztp-types:cmp-csr from module ietf-ztp-csr and offer both proof-of-possession and proof-of-identity. To allow PKI management entities that use the module ietf-ztp-csr and also wish to comply with this profile, the ir, cr, kur, or p10cr message **MUST** be formatted by the EE as described in Section 4.1, and it **MAY** be forwarded, as specified in Section 5.2.

In Bootstrapping Remote Secure Key Infrastructure (BRSKI) [RFC8995] environments, "BRSKI-AE: Alternative Enrollment Protocols in BRSKI" [BRSKI-AE] describes a generalization regarding the employed enrollment protocols to allow alternatives to Enrollment over Secure Transport (EST) [RFC7030]. For the use of CMP, it requires adherence to this profile.

1.8. Scope of This Document

On one hand, this profile intends to reduce the flexibility of CMP to the generic needs of automated certificate management of machine end entities. On the other hand, it offers a variety of PKI management operations and options relevant for industrial use cases. Therefore, it is still a framework that supports further profiling by those addressing a specific use case or scenario, e.g., 3GPP/ETSI or UNISIG. There is room to further tailor this profile. This enables stricter profiling to meet the concrete needs in application areas.

To minimize ambiguity and complexity through needless variety, this document specifies exhaustive requirements for generating PKI management messages on the sender side. However, it gives only minimal requirements on checks by the receiving side and how to handle error cases.

Especially on the EE side, this profile aims at a lightweight implementation. This means that the number of PKI management operation implementations are reduced to a reasonable minimum to support typical certificate management use cases in industrial machine-to-machine environments. On the EE side, only limited resources are expected, while on the side of the PKI management entities, the profile accepts higher requirements.

For the sake of interoperability and robustness, implementations should, so long as security is not affected, adhere to Postel's law: "Be conservative in what you do, be liberal in what you accept from others" (often reworded as: "Be conservative in what you send, be liberal in what you receive").

Fields used in ASN.1 syntax in Sections 3, 4, or 5 are specified in CMP [RFC4210] [RFC9480], CRMF [RFC4211], and CMS [RFC5652] [RFC8933]. When these sections do not explicitly discuss a field, then the field **SHOULD NOT** be used by the sending entity. The receiving entity **MUST NOT** require the absence of such a field and, if the field is present, **MUST** handle it gracefully.

1.9. Structure of This Document

Section 2 introduces the general PKI architecture and approach to certificate management that is assumed in this document.

Section 3 profiles the generic aspects of the PKI management operations specified in detail in Sections 4 and 5 to minimize redundancy in the description and to ease implementation. This covers the general structure and protection of messages, as well as generic prerequisites, validation, and error handling.

Section 4 profiles the exchange of CMP messages between an EE and the PKI management entity. There are various flavors of certificate enrollment requests, optionally with polling, central key generation, revocation, and general support PKI management operations.

Section 5 profiles responding to requests, exchanges between PKI management entities, and operations on behalf of other PKI entities. This may include delayed delivery of messages, which involves polling for responses, and nesting of messages.

Section 6 outlines several mechanisms for CMP message transfer, including HTTP-based transfer [RFC6712] optionally using TLS, CoAP-based transfer [RFC9482] optionally using DTLS, and offline file-based transport.

Section 7 defines which parts of the profile are mandatory, recommended, optional, or not relevant to implement for which type of entity.

2. Solution Architecture

To facilitate secure automatic certificate enrollment, the device hosting an EE is typically equipped with a manufacturer-issued device certificate. Such a certificate is typically installed during production and is meant to identify the device throughout its lifetime. This certificate can be used to protect the initial enrollment of operational certificates after installation of the EE in its operational environment. In contrast to the manufacturer-issued device certificate, operational certificates are issued by the owner or operator of the device to identify the device or one of its components for operational use, e.g., in a security protocol like IPsec, TLS, or SSH. In IEEE 802.1AR [IEEE.802.1AR_2018], a manufacturer-issued device certificate is called an Initial Device Identifier (IDevID) certificate and an operational certificate is called a Locally Significant Device Identifier (LDevID) certificate.

Note: The owner or operator using the manufacturer-issued device certificate for authenticating the device during initial enrollment of operational certificates **MUST** trust the respective trust anchor provided by the manufacturer.

Note: According to IEEE 802.1AR [IEEE.802.1AR_2018], a DevID comprises the triple of the certificate, the corresponding private key, and the certificate chain.

All certificate management operations specified in this document follow the pull model, i.e., they are initiated by an EE (or by an RA acting as an EE). The EE creates a CMP request message, protects it using some asymmetric credential or shared secret information, and sends it to a PKI management entity. This PKI management entity may be a CA or more typically an RA, which checks the request and responds to it itself or forwards the request upstream to the next PKI management entity. In case an RA changes the CMP request message header or body or wants to demonstrate successful verification or authorization, it can apply a protection of its own. The communication between an LRA and RA can be performed synchronously or asynchronously. Asynchronous communication typically leads to delayed message delivery as described in Section 4.4.

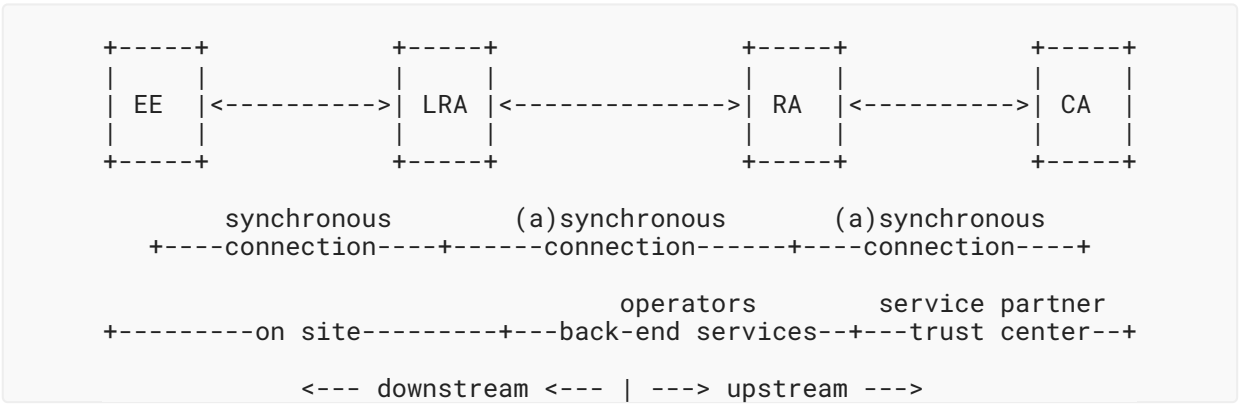


Figure 1: Certificate Management Architecture Example

In operational environments, the certificate management architecture can have multiple LRAs bundling requests from multiple EEs at dedicated locations and one (or more than one) central RA aggregating the requests from the LRAs. Every LRA in this scenario has shared secret information (one per EE) for MAC-based protection or a CMP protection key and certificate, allowing it to protect CMP messages it processes using its own credentials. The figure above shows an architectural example with one LRA, RA, and CA. It is also possible not to have an RA or LRA or that there is no CA with a CMP interface. Depending on the network infrastructure, the message transfer between PKI management entities may be based on synchronous online connections, asynchronous connections, or even offline (e.g., file-based) transfer.

Note: In contrast to the pull model used in this document, other specifications could use the messages specified in this document to implement the push model. In this case, the EE is pushed (triggered) by the PKI management entity to provide the CMP request; therefore, the EE acts as the receiver, not initiating the interaction with the PKI. For example, when the device itself only acts (as a server as described in BRSKI with Pledge in Responder Mode [BRSKI-PRM]), support of certificate enrollment in a push model is needed. While BRSKI-PRM currently utilizes its own format for the exchanges, CMP in general and the messages specified in this profile offer all required capabilities. Nevertheless, the message flow and state machine as described in Section 4 must be adapted to implement a push model.

Note: Third-party CAs not conforming to this document may implement other variants of CMP, different standardized protocols, or even proprietary interfaces for certificate management. In such cases, an RA needs to adapt the exchanged CMP messages to the flavor of certificate management interaction required by such a nonconformant CA.

3. Generic Aspects of PKI Messages and PKI Management Operations

This section covers the generic aspects of the PKI management operations specified in Sections 4 and 5 as upfront general requirements to minimize redundancy in the description and to ease implementation.

As described in Section 5.1 of [RFC4210], all CMP messages have the following general structure:

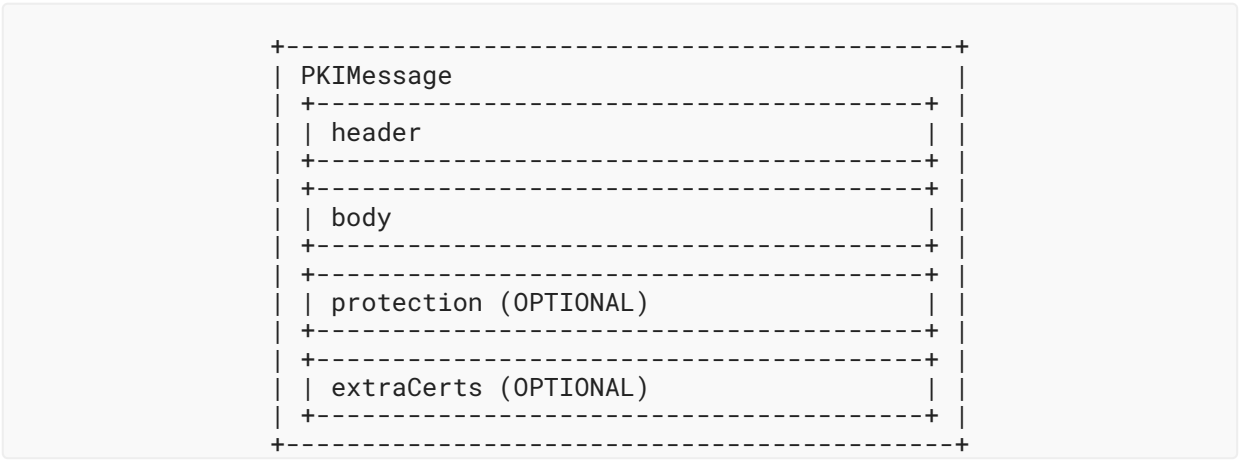


Figure 2: CMP Message Structure

The general contents of the message header, protection, and extraCerts fields are specified in the following three subsections.

In case a specific PKI management operation needs different contents in the header, protection, or extraCerts fields, the differences are described in the respective subsections of Sections 4 and 5.

The CMP message body contains the PKI management operation-specific information. It is described in Sections 4 and 5.

Note: In the description of CMP messages, the presence of some fields is stated as **OPTIONAL** or **RECOMMENDED**. The following text that states requirements on such a field applies only if the field is present.

The generic prerequisites needed by the PKI entities in order to perform PKI management operations are described in Section 3.4.

The generic validation steps to be performed by PKI entities upon receiving a CMP message are described in Section 3.5.

The generic aspects of handling and reporting errors are described in Section 3.6.

3.1. General Description of the CMP Message Header

This section describes the generic header fields of all CMP messages.

Any fields or variations specific to PKI management operation are described in Sections 4 and 5.

```
header
  pvno                        REQUIRED
    -- MUST be 3 to indicate CMP v3 in all cases where EnvelopedData
    --   is supported and expected to be used in the current
    --   PKI management operation
    -- MUST be 3 to indicate CMP v3 in certConf messages when using
    --   the hashAlg field
    -- MUST be 2 to indicate CMP v2 in all other cases
    -- For details on version negotiation, see [RFC9480]
  sender                      REQUIRED
    -- Contains a name representing the originator, which also
    --   protects the message
    -- For signature-based protection, MUST be the subject field of
    --   the CMP protection certificate
    -- For MAC-based protection, MUST contain a name the PKI
    --   management entity can use to identify the shared secret
    --   information. This name MUST be placed in the commonName
    --   field of the directoryName choice.
    -- In a multihop scenario, the receiving entity cannot rely
    --   on the correctness of the sender field.
  recipient                   REQUIRED
    -- SHOULD be the name of the intended recipient; otherwise, the
    --   NULL-DN MUST be used
    -- In the first message of a PKI management operation, SHOULD be
    --   the subject DN of the CA the PKI management operation is
    --   requested from
    -- In all other messages, SHOULD contain the value of the sender
    --   field of the previous message in the same PKI management
    --   operation
    -- The recipient field shall be handled gracefully by the
    --   receiving entity, because in a multihop scenario, its
    --   correctness cannot be guaranteed.
  messageTime                 OPTIONAL
    -- MUST be present if the confirmWaitTime field is present
    -- MUST be the time at which the message was produced, if present
    -- MAY be set by a PKI management entity to provide the current
    --   time
    -- MAY be used by the end entity for time synchronization if the
    --   response was received within a short time frame
  protectionAlg               REQUIRED
    -- MUST be an algorithm identifier indicating the algorithm
    --   used for calculating the protection bits
    -- If it is a signature algorithm, its type MUST be
    --   MSG_SIG_ALG as specified in Section 3 of [RFC9481] and
    --   MUST be consistent with the subjectPublicKeyInfo field of
```

```
    --   the CMP protection certificate
    -- If it is a MAC algorithm, its type MUST be MSG_MAC_ALG, as
    --   specified in [RFC9481], Section 6.1
  senderKID                  RECOMMENDED
    -- For signature-based protection, MUST be used and contain the
    --   value of the SubjectKeyIdentifier if present in the CMP
    --   protection certificate
    -- For MAC-based protection, MUST be used and contain the same
    --   name as in the commonName field of the sender field
  transactionID              REQUIRED
    -- In the first message of a PKI management operation, MUST be
    --   128 bits of random data to minimize the probability of
    --   having the transactionID already in use at the server
    -- In all other messages, MUST be the value from the previous
    --   message in the same PKI management operation
  senderNonce                REQUIRED
    -- MUST be cryptographically secure and fresh 128 random bits
  recipNonce                 RECOMMENDED
    -- If this is the first message of a transaction, MUST be absent
    -- If this is a delayed response message, MUST be present and
    --   contain the value of the senderNonce of the respective
    --   request message in the same transaction
    -- In all other messages, MUST be present and contain the value
    --   of the senderNonce of the previous message in the same
    --   transaction
  generalInfo                OPTIONAL
    implicitConfirm          OPTIONAL
    -- RECOMMENDED in ir/cr/kur/p10cr messages,
    -- OPTIONAL in ip/cp/kup response messages, and
    -- PROHIBITED in other types of messages
    -- Added to request messages to request omission of the certConf
    --   message
    -- Added to response messages to grant omission of the certConf
    --   message
    -- See [RFC4210], Section 5.1.1.1.
      ImplicitConfirmValue   REQUIRED
    -- ImplicitConfirmValue MUST be NULL
    confirmWaitTime          OPTIONAL
    -- RECOMMENDED in ip/cp/kup messages if implicitConfirm is
    --   not included
    -- PROHIBITED if implicitConfirm is included
    -- See [RFC4210], Section 5.1.1.2.
      ConfirmWaitTimeValue   REQUIRED
    -- ConfirmWaitTimeValue MUST be a GeneralizedTime value
    --   specifying the point in time up to which the PKI management
    --   entity will wait for the certConf message.  The accepted
    --   length of the waiting period will vary by use case.
    certProfile              OPTIONAL
    -- MAY be present in ir/cr/kur/p10cr and in genm messages of type
    --   id-it-certReqTemplate
    -- MUST be omitted in all other messages
    -- See [RFC9480].
      CertProfileValue       REQUIRED
    -- MUST contain a sequence of one UTF8String element
    -- MUST contain the name of a certificate profile
```

3.2. General Description of the CMP Message Protection

This section describes the generic protection field contents of all CMP messages. For signature-based protection, which is the default protection mechanism for all CMP messages described in this profile, the CMP protection key and CMP protection certificate are used. For MAC-based protection, shared secret information is used as described in Section 4.1.5.

```
protection
    -- If present, the same kind of protection MUST be used for all
    --   messages of that PKI management operation.
    -- MUST be present, except if protection is not possible for
    --   error messages as described in Section 3.6.4
    -- For signature-based protection, MUST contain the signature
    --   calculated using the CMP protection key of the entity
    --   protecting the message
    -- For MAC-based protection, MUST contain a MAC calculated using
    --   the shared secret information
    -- The protection algorithm used MUST be given in the
    --   protectionAlg field.
```

The CMP message protection provides, if available, message origin authentication and integrity protection for the header and body. The CMP message extraCerts field is not covered by this protection.

Note: The extended key usages described in Section 2.2 of CMP Updates [RFC9480] can be used for authorization of a sending PKI management entity.

3.3. General Description of CMP Message ExtraCerts

This section describes the generic extraCerts field of all CMP messages. Any specific requirements on the extraCerts are specified in the respective PKI management operation.

```
extraCerts
    -- MUST be present for signature-based protection and contain the
    --   CMP protection certificate together with its chain for the
    --   first request and response message of a PKI management
    --   operation.  MAY be omitted in certConf, PKIConf, pollReq,
    --   and pollRep messages.  The first certificate in this field
    --   MUST be the CMP protection certificate followed by its
    --   chain, where each element should directly certify the one
    --   immediately preceding it.
    -- MUST be present in ip, cp, and kup messages and contain the
    --   chain of a newly issued certificate.
    -- Self-signed certificates should be omitted from extraCerts and
    --   MUST NOT be trusted based on their inclusion in any case
```

Note: One reason for adding a self-signed certificate to extraCerts is if it is the CMP protection certificate or a successor root CA self-signed certificate as indicated in the HashOfRootKey extension of the current root CA certificate; see [RFC8649]. Another reason for including self-

signed certificates in the extraCerts is, for instance, due to storage limitations. A receiving PKI entity may not have the complete trust anchor information available but just a unique identification of it and thus needs the full trust anchor information carried in a self-signed certificate for further processing (see Section 9).

For maximum interoperability, all implementations **SHOULD** be prepared to handle potentially additional certificates and arbitrary orderings of the certificates.

3.4. Generic PKI Management Operation Prerequisites

This subsection describes what is generally needed by the PKI entities to be able to perform PKI management operations.

Identification of PKI entities:

- For signature-based protection, each EE knows its own identity from the CMP protection certificate; for MAC-based protection, it **MAY** know its identity to fill the sender field.
- Each EE **MAY** know the intended recipient of its requests to fill the recipient field, e.g., the name of the addressed CA.

  Note: This name may be established using an enrollment voucher (as described in [RFC8366]), the issuer field from a CertReqTemplate response message content, or by other configuration means.

Routing of CMP messages:

- Each PKI entity sending messages upstream **MUST** know the address needed for transferring messages to the next PKI management entity in case online transfer is used.

  Note: This address may depend on the recipient, the certificate profile, and the used transfer mechanism.

Authentication of PKI entities:

- Each PKI entity **MUST** have credentials to authenticate itself. For signature-based protection, it **MUST** have a private key and the corresponding certificate along with its chain.
- Each PKI entity **MUST** be able to establish trust in the PKI it receives responses from. When signature-based protection is used, it **MUST** have the trust anchor(s) and any certificate status information needed to perform path validation of CMP protection certificates used for signature-based protection.

  Note: A trust anchor is usually a root certificate of the PKI addressed by the requesting EE. It may be established by configuration or in an out-of-band manner. For an EE, it may be established using an enrollment voucher [RFC8366] or in-band of CMP by the caPubs field in a certificate response message.

Authorization of PKI management operations:

- Each EE or RA **MUST** have sufficient information to be able to authorize the PKI management entity to perform the upstream PKI management operation.

Note: This may be achieved, for example, by using the cmcRA extended key usage in server certificates, by local configuration (such as specific name patterns for subject Distinguished Name (DN) or Subject Alternative Name (SAN) portions that may identify an RA) and/or by having a dedicated root CA usable only for authenticating PKI management entities.

- Each PKI management entity **MUST** have sufficient information to be able to authorize the downstream PKI entity requesting the PKI management operation.

  Note: For authorizing an RA, the same examples apply as above. The authorization of EEs can be very specific to the application domain based on local PKI policy.

3.5. Generic Validation of a PKI Message

This section describes generic validation steps of each PKI entity receiving a PKI request or response message before any further processing or forwarding. If a PKI management entity decides to terminate a PKI management operation because a check failed, it **MUST** send a negative response or an error message as described in Section 3.6. The PKIFailureInfo bits given below in parentheses **MAY** be used in the failInfo field of the PKIStatusInfo as described in Section 3.6.4; also see Appendix F of [RFC4210].

All PKI message header fields not mentioned in this section, like the recipient and generalInfo fields, **SHOULD** be handled gracefully upon receipt.

The following list describes the basic set of message input validation steps. Without these checks, the protocol becomes dysfunctional.

- The formal ASN.1 syntax of the whole message **MUST** be compliant with the definitions given in CMP [RFC4210] [RFC9480], CRMF [RFC4211], and CMS [RFC5652] [RFC8933]. (failInfo: badDataFormat)
- The pvno **MUST** be cmp2000(2) or cmp2021(3). (failInfo bit: unsupportedVersion)
- The transactionID **MUST** be present. (failInfo bit: badDataFormat)
- The PKI message body type **MUST** be one of the message types supported by the receiving PKI entity and **MUST** be allowed in the current state of the PKI management operation identified by the given transactionID. (failInfo bit: badRequest)

The following list describes the set of message input validation steps required to ensure secure protocol operation:

- The senderNonce **MUST** be present and **MUST** contain at least 128 bits of data. (failInfo bit: badSenderNonce)
- Unless the PKI message is the first message of a PKI management operation,
  - the recipNonce **MUST** be present and **MUST** equal the senderNonce of the previous message or equal the senderNonce of the most recent request message for which the response was delayed, in case of delayed delivery as specified in Section 4.4. (failInfo bit: badRecipientNonce)
- Messages without protection **MUST** be rejected except for error messages as described in Section 3.6.4.

- The message protection **MUST** be validated when present, and messages with an invalid protection **MUST** be rejected.
  - The protection **MUST** be signature-based except if MAC-based protection is used as described in Sections 4.1.5 and 4.1.6.3. (failInfo bit: wrongIntegrity)
  - If present, the senderKID **MUST** identify the key material needed for verifying the message protection. (failInfo bit: badMessageCheck)
  - If signature-based protection is used, the CMP protection certificate **MUST** be successfully validated, including path validation using a trust anchor, and **MUST** be authorized according to local policies. If the keyUsage extension is present in the CMP protection certificate, the digitalSignature bit **MUST** be set. (failInfo bit: badAlg, badMessageCheck, or signerNotTrusted)
  - The sender of a request message **MUST** be authorized to request the operation according to PKI policies. (failInfo bit: notAuthorized)

Note: The requirements for checking certificates given in [RFC5280] **MUST** be followed for signature-based CMP message protection. Unless the message is a positive ip/cp/kup, where the issuing CA certificate of the newly enrolled certificate is the same as the CMP protection certificate of that message, certificate status checking **SHOULD** be performed on the CMP protection certificates. If the response message contains the caPubs field to transfer new trust anchor information, the CMP protection is crucial and certificate status checking is **REQUIRED**. For other cases, it **MAY** be acceptable to omit certificate status checking when respective information is not available.

Depending on local policies, one or more of the input validation checks described below need to be implemented:

- If signature-based protection is used, the sender field **MUST** match the subject of the CMP protection certificate. (failInfo bit: badMessageCheck)
- If the messageTime is present and
  - the receiving system has a reliable system time, the messageTime **MUST** be close to the current time of the receiving system, where the threshold will vary by use case. (failInfo bit: badTime)
  - the receiving system does not have a reliable system time, the messageTime **MAY** be used for time synchronization.

3.6. Error Handling

This section describes how a PKI entity handles error conditions on messages it receives. Each error condition should be logged appropriately to allow root-cause analysis of failure cases.

3.6.1. Reporting Error Conditions Upstream

An EE **SHALL NOT** send error messages. PKI management entities **SHALL NOT** send error messages in the upstream direction either.

In case an EE rejects a newly issued certificate contained in an ip, cp, or kup message and implicit confirmation has not been granted, the EE **MUST** report this using a certConf message with "rejection" status and await the pkiConf response as described in Section 4.1.1.

On all other error conditions regarding response messages, the EE or PKI management entity **MUST** regard the current PKI management operation as terminated with failure. The error conditions include:

- invalid response message header, body type, protection, or extraCerts, according to the checks described in Section 3.5,
- any issue detected with response message contents,
- receipt of an error message from upstream,
- timeout occurred while waiting for a response, and
- rejection of a newly issued certificate while implicit confirmation has been granted.

Upstream PKI management entities will not receive any CMP message to learn that the PKI management operation has been terminated. In case they expect a further message from the EE, a connection interruption or timeout will occur. The value set for such timeouts will vary by use case. Then they **MUST** also regard the current PKI management operation as terminated with failure and **MUST NOT** attempt to send an error message downstream.

3.6.2. Reporting Error Conditions Downstream

In case the PKI management entity detects an error condition, e.g., rejecting the request due to policy decision, in the body of an ir, cr, p10cr, kur, or rr message received from downstream, it **MUST** report the error in the specific response message, i.e., an ip, cp, kup, or rp with "rejection" status, as described in Sections 4.1.1 and 4.2. This can also happen in case of polling.

In case the PKI management entity detects any other error condition on requests (including pollReq, certConf, genm, and nested messages) received from downstream and on responses received from upstream (such as invalid message header, body type, protection, or extraCerts, according to the checks described in Section 3.5), it **MUST** report them downstream in the form of an error message as described in Section 3.6.4.

3.6.3. Handling Error Conditions on Nested Messages Used for Batching

Batching of messages using nested messages as described in Section 5.2.2.2 requires special error handling.

If the error condition is on an upstream nested message containing batched requests, it **MUST NOT** attempt to respond to the individual requests included in it but to the nested message itself.

In case a PKI management entity receives an error message in response to a nested message, it must propagate the error by responding with an error message to each of the request messages contained in the nested message.

In case a PKI management entity detects an error condition on the downstream nested message received in response to a nested message sent before and the body of the received nested message still parses, it **MAY** ignore this error condition and handle the included responses as described in Section 5.2.2.2. Otherwise, it **MUST** propagate the error by responding with an error message to each of the requests contained in the nested message it sent originally.

3.6.4. PKIStatusInfo and Error Messages

When sending any kind of negative response, including error messages, a PKI entity **MUST** indicate the error condition in the PKIStatusInfo structure of the respective message as described below. Then it **MUST** regard the current PKI management operation as terminated with failure.

The PKIStatusInfo structure is used to report errors. It may be part of various message types, in particular, ip, cp, kup, certConf, and error. The PKIStatusInfo structure consists of the following fields:

status: Here, the PKIStatus value "rejection" **MUST** be used in case an error was detected. When a PKI management entity indicates delayed delivery of a CMP response message to the EE with an error message as described in Section 4.4, the status "waiting" **MUST** be used there.

statusString: Here, any human-readable valid value for logging or to display via a user interface should be added.

failInfo: Here, the PKIFailureInfo bits **MAY** be used in the way explained in Appendix F of [RFC4210]. PKIFailureInfo bits regarding the validation described in Section 3.5 are referenced there. The PKIFailureInfo bits referenced in Sections 5.1 and 6 are described here:

badCertId: A kur, certConf, or rr message references an unknown certificate.

badPOP: An ir/cr/kur/p10cr contains an invalid proof-of-possession.

certRevoked: Revocation is requested for a certificate that is already revoked.

badCertTemplate: The contents of a certificate request are not accepted, e.g., a field is missing or has an unacceptable value or the given public key is already in use in some other certificate (depending on policy).

transactionIdInUse: This is sent by a PKI management entity in case the received request contains a transactionID that is currently in use for another transaction. An EE receiving such an error message should resend the request in a new transaction using a different transactionID.

notAuthorized: The sender of a request message is not authorized for requesting the operation.

systemUnavail: This is sent by a PKI management entity in case a back-end system is not available.

systemFailure: This is sent by a PKI management entity in case a back-end system is currently not functioning correctly.

An EE receiving a systemUnavail or systemFailure failInfo should resend the request in a new transaction after some time.

Detailed Message Description:

```
Error Message -- error

Field                          Value

header
    -- As described in Section 3.1

body
    -- The message indicating the error that occurred
  error                        REQUIRED
    pKIStatusInfo              REQUIRED
      status                   REQUIRED
    -- MUST have the value "rejection"
      statusString             OPTIONAL
    -- This field should contain any human-readable text for
    --   debugging, for logging, or to display in a GUI
      failInfo                 OPTIONAL
    -- MAY be present and contain the relevant PKIFailureInfo bits

protection                     RECOMMENDED
    -- As described in Section 3.2

extraCerts                     RECOMMENDED
    -- As described in Section 3.3
```

Protecting the error message may not be technically feasible if it is not clear which credential the recipient will be able to use when validating this protection, e.g., in case the request message was fundamentally broken. In these exceptional cases, the protection of the error message **MAY** be omitted.

4. PKI Management Operations

This section focuses on the communication of an EE with the PKI management entity it directly talks to. Depending on the network and PKI solution, this can be an RA or directly a CA. Handling of a message by a PKI management entity is described in Section 5.

The PKI management operations specified in this section cover the following:

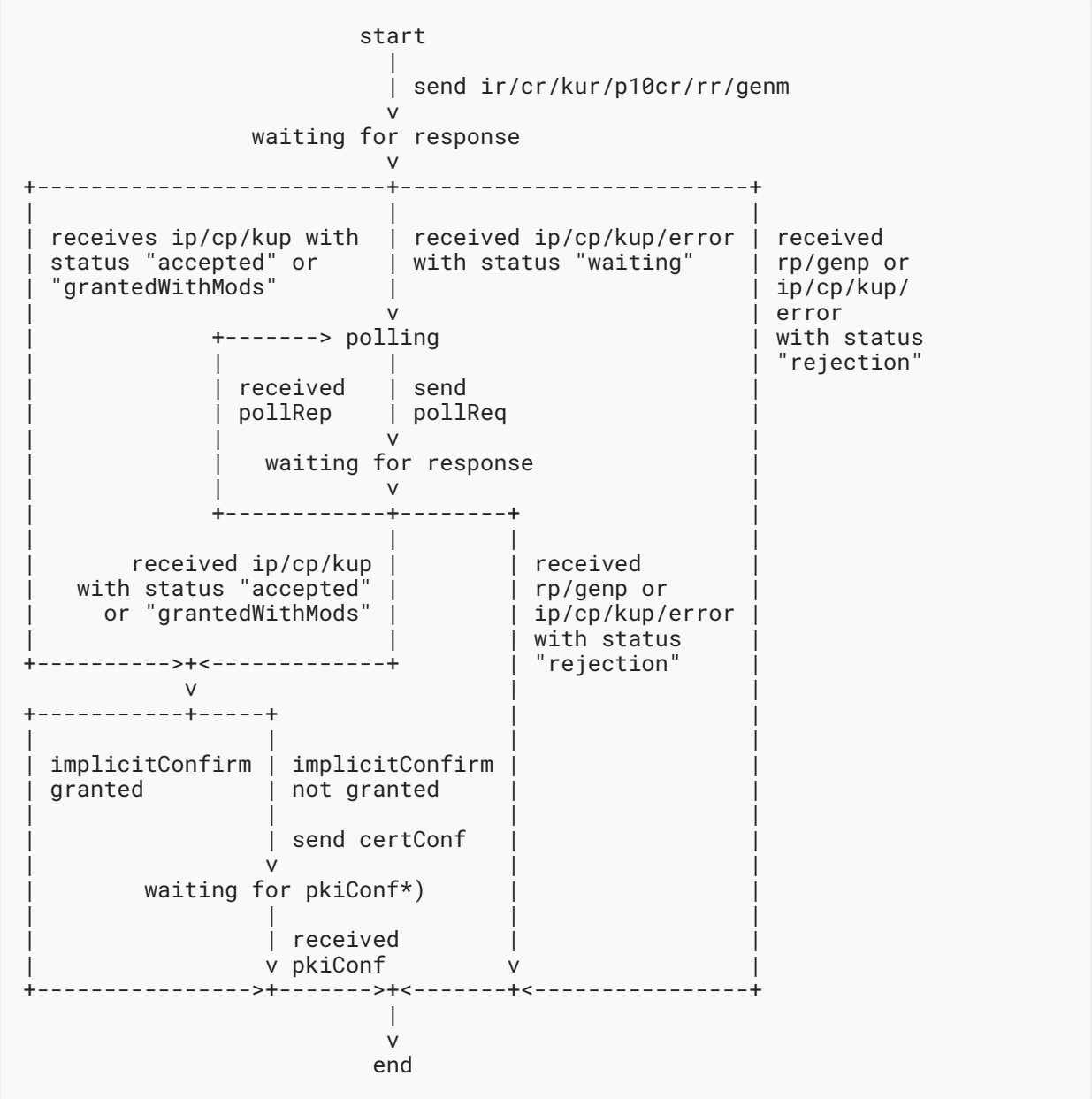
- requesting a certificate with variations like initial enrollment, certificate updates, central key generation, and MAC-based protection
- revoking a certificate
- support messages
- polling for delayed response messages

These operations mainly specify the message body of the CMP messages and utilize the specification of the message header, protection, and extraCerts, as specified in Section 3. The messages are named by the respective field names in PKIBody, like ir, ip, cr, cp, etc.; see Section 5.1.2 of [RFC4210].

The following diagram shows the EE state machine covering all PKI management operations described in this section, including negative responses, error messages described in Section 3.6.4, ip/cp/kup/error messages with status "waiting", and pollReq and pollRep messages as described in Section 4.4.

On receiving messages from upstream, the EE **MUST** perform the general validation checks described in Section 3.5. In case an error occurs, the behavior is described in Section 3.6.

End Entity State Machine:



*) In case of a delayed delivery of pkiConf responses, the same polling mechanism is initiated as for rp or genp messages by sending an error message with status "waiting".

Note: All CMP messages belonging to the same PKI management operation **MUST** have the same transactionID because the message receiver identifies the elements of the operation in this way.

This section is aligned with CMP [RFC4210], CMP Updates [RFC9480], and CMP Algorithms [RFC9481].

Guidelines as well as an algorithm use profile for this document are available in CMP Algorithms [RFC9481].

4.1. Enrolling End Entities

There are various approaches for requesting a certificate from a PKI.

These approaches differ in the way the EE authenticates itself to the PKI, in the form of the request being used, and how the key pair to be certified is generated. The authentication mechanisms may be as follows:

- using a certificate from an external PKI, e.g., a manufacturer-issued device certificate, and the corresponding private key
- using a private key and certificate issued from the same PKI that is addressed for requesting a certificate
- using the certificate to be updated and the corresponding private key
- using shared secret information known to the EE and the PKI management entity

An EE requests a certificate indirectly or directly from a CA. When the PKI management entity handles the request as described in Section 5.1.1 and responds with a message containing the requested certificate, the EE **MUST** reply with a confirmation message unless implicitConfirm was granted. The PKI management entity **MUST** then handle it as described in Section 5.1.2 and respond with a confirmation, closing the PKI management operation.

The message sequences described in this section allow the EE to request certification of a locally or centrally generated public-private key pair. The public key and the subject name identifying the EE **MUST** be present in the certTemplate of the certificate request message.

Note: If the EE does not know for which subject name to request the certificate, it can use the subject name from the CMP protection certificate in case of signature-based protection or the identifier of the shared secret in case of MAC-based protection.

Typically, the EE provides a signature-based proof-of-possession of the private key associated with the public key contained in the certificate request, as defined by [RFC4211], Section 4.1, clause 3. To this end, it is assumed that the private key can technically be used for signing. This is the case for the most common algorithms RSA, ECDSA, and EdDSA, regardless of potentially intended restrictions of the key usage.

Note: Section 4 of [RFC4211] allows for providing proof-of-possession using any method that a key can be used for. In conformance with Section 8.1.5.1.1.2 of [NIST.SP.800-57p1r5], the newly generated private key may be used for self-signature, if technically possible, even if the keyUsage extension requested in the certificate request prohibits generation of digital signatures.

The requesting EE provides the binding of the proof-of-possession to its identity by signature-based or MAC-based protection of the CMP request message containing that POP. An upstream PKI management entity should verify whether this EE is authorized to obtain a certificate with the requested subject and other fields and extensions.

The proof-of-possession is provided by signing the certReq containing the certTemplate with the subject name and public key. To bind this proof-of-possession to the proof-of-identity of the requesting EE, the subject name in the certTemplate needs to identify the same entity as the subject name in the CMP protection certificate or match the identifier used with MAC-based protection.

Note: This binding may be lost if a PKI management entity reprotects this request message.

The EE **MAY** indicate the certificate profile to use in the certProfile extension of the generalInfo field in the PKIHeader of the certificate request message as described in Section 3.1.

In case the EE receives a CA certificate in the caPubs field for installation as a new trust anchor, it **MUST** properly authenticate the message and authorize the sender as a trusted source of the new trust anchor. This authorization is typically indicated using shared secret information for protecting an Initialization Response (ip) message. Authorization can also be signature-based, using a certificate issued by another PKI that is explicitly authorized for this purpose. A certificate received in caPubs **MUST NOT** be accepted as a trust anchor if it is the root CA certificate of the certificate used for protecting the message.

4.1.1. Enrolling an End Entity to a New PKI

This PKI management operation should be used by an EE to request a certificate from a new PKI using an existing certificate from an external PKI, e.g., a manufacturer-issued IDevID certificate [IEEE.802.1AR_2018], to authenticate itself to the new PKI.

Note: In Bootstrapping Remote Secure Key Infrastructure (BRSKI) [RFC8995] environments, "BRSKI-AE: Alternative Enrollment Protocols in BRSKI" [BRSKI-AE] describes a generalization regarding enrollment protocols alternative to EST [RFC7030]. As replacement of EST simpleenroll, BRSKI-AE uses this PKI management operation for bootstrapping LDevID certificates.

Specific prerequisites augmenting the prerequisites in Section 3.4 are as follows:

- The certificate of the EE **MUST** have been enrolled by an external PKI, e.g., a manufacturer-issued device certificate.
- The PKI management entity **MUST** have the trust anchor of the external PKI.
- When using the generalInfo field certProfile, the EE **MUST** know the identifier needed to indicate the requested certificate profile.

Message Flow:

```
Step# EE                                    PKI management entity
  1   format ir
  2                       ->   ir      ->
  3                                         handle or
                                              forward ir
  4                                         format or receive ip
  5                                         possibly grant
                                              implicitConfirm
  6                       <-   ip      <-
  7   handle ip

-----------------  if implicitConfirm not granted  -----------------

  8   format certConf
  9                       ->   certConf ->
 10                                         handle or
                                              forward certConf
 11                                         format or receive pkiConf
 12                       <-   pkiConf  <-
 13   handle pkiConf
```

For this PKI management operation, the EE **MUST** include a sequence of one CertReqMsg in the ir. If more certificates are required, further requests **MUST** be sent using separate PKI management operations.

The EE **MUST** include the generalInfo field implicitConfirm in the header of the ir message as described in Section 3.1, unless it requires certificate confirmation. This leaves the PKI management entities the choice of whether or not the EE must send a certConf message upon receiving a new certificate. Depending on the PKI policy and requirements for managing EE certificates, it can be important for PKI management entities to learn if the EE accepted the new certificate. In such cases, when responding with an ip message, the PKI management entity **MUST NOT** include the implicitConfirm extension. In case the EE included the generalInfo field implicitConfirm in the request message and the PKI management entity does not need any explicit confirmation from the EE, the PKI management entity **MUST** include the generalInfo field implicitConfirm in the response message. This prevents explicit certificate confirmation and saves the overhead of a further message round trip. Otherwise, the PKI management entity **SHOULD** include confirmWaitTime as described in Section 3.1.

If the EE did not request implicit confirmation or implicit confirmation was not granted by the PKI management entity, certificate confirmation **MUST** be performed as follows. If the EE successfully received the certificate, it **MUST** send a certConf message in due time. On receiving a valid certConf message, the PKI management entity **MUST** respond with a pkiConf message. If the PKI management entity does not receive the expected certConf message in time, it **MUST** handle this like a rejection by the EE. In case of rejection, depending on its policy, the PKI management entity **MAY** revoke the newly issued certificate, notify a monitoring system, or log the event internally.

Note: Depending on PKI policy, a new certificate may be published by a PKI management entity, and explicit confirmation may be required. In this case, it is advisable not to do the publication until a positive certificate confirmation has been received. This way, the need to revoke the certificate on negative confirmation can be avoided.

If the certificate request was rejected by the CA, the PKI management entity **MUST** return an ip message containing the status code "rejection" as described in Section 3.6, and the certifiedKeyPair field **SHALL** be omitted. The EE **MUST NOT** react to such an ip message with a certConf message, and the PKI management operation **MUST** be terminated.

Detailed Message Description:

```
Initialization Request -- ir

Field                         Value

header
    -- As described in Section 3.1

body
    -- The request of the EE for a new certificate
  ir                          REQUIRED
    -- MUST contain a sequence of one CertReqMsg
    -- If more certificates are required, further PKI management
    --   operations needs to be initiated
    certReq                   REQUIRED
      certReqId               REQUIRED
    -- MUST be 0
      certTemplate            REQUIRED
        version               OPTIONAL
    -- MUST be 2 if supplied
        subject               REQUIRED
    -- The EE's identity MUST be carried in the subject field
    --   and/or the subjectAltName extension.
    -- If subject name is present only in the subjectAltName
    --   extension, then the subject field MUST be NULL-DN
       publicKey              OPTIONAL
    -- MUST be present if local key generation is used
    -- MAY be absent if central key generation is requested
         algorithm            OPTIONAL
    -- MUST be present if local key generation is used and MUST
    --   include the subject public key algorithm identifier
    -- MAY be present if central key generation is requested and,
    --   if present, informs the KGA of algorithm and parameter
    --   preferences regarding the to-be-generated key pair
        subjectPublicKey      REQUIRED
    -- MUST contain the public key to be certified in case of local
    --   key generation
    -- MUST be a zero-length BIT STRING if central key generation
    --   is requested
       extensions             OPTIONAL
    -- MAY include end-entity-specific X.509 extensions of the
    --   requested certificate, like subject alternative name, key
    --   usage, and extended key usage
    -- The subjectAltName extension MUST be present if the EE subject
```

```
    --   name includes a subject alternative name.
    popo                      OPTIONAL
    -- MUST be present if local key generation is used
    -- MUST be absent if central key generation is requested
      signature               OPTIONAL
    -- MUST be used by an EE if the key can be used for signing, and
    --   if used, it MUST have the type POPOSigningKey
       poposkInput            PROHIBITED
    -- MUST NOT be used; it is not needed because subject and
    --   publicKey are both present in the certTemplate
       algorithmIdentifier    REQUIRED
    -- The signature algorithm MUST be consistent with the publicKey
    --   algorithm field of the certTemplate
       signature              REQUIRED
    -- MUST contain the signature value computed over the DER-encoded
    --   certReq
      raVerified              OPTIONAL
    -- MAY be used by an RA after verifying the proof-of-possession
    --   provided by the EE

protection                    REQUIRED
    -- As described in Section 3.2

extraCerts                    REQUIRED
    -- As described in Section 3.3


Initialization Response -- ip

Field                         Value

header
    -- As described in Section 3.1

body
    -- The response of the CA to the request as appropriate
  ip                          REQUIRED
    caPubs                    OPTIONAL
    -- MAY be used if the certifiedKeyPair field is present
    -- If used, it MUST contain only a trust anchor, e.g., root
    --   certificate, of the certificate contained in certOrEncCert
    response                  REQUIRED
    -- MUST contain a sequence of one CertResponse
      certReqId              REQUIRED
    -- MUST be 0
      status                  REQUIRED
    -- PKIStatusInfo structure MUST be present
       status                 REQUIRED
    -- positive values allowed: "accepted", "grantedWithMods"
    -- negative values allowed: "rejection"
    -- "waiting" only allowed with a polling use case as described
    --   in Section 4.4
       statusString           OPTIONAL
    -- MAY be any human-readable text for debugging, for logging, or
    --   to display in a GUI
       failInfo               OPTIONAL
    -- MAY be present if status is "rejection"
    -- MUST be absent if status is "accepted" or "grantedWithMods"
```

```
      certifiedKeyPair        OPTIONAL
    -- MUST be present if status is "accepted" or "grantedWithMods"
    -- MUST be absent if status is "rejection"
       certOrEncCert          REQUIRED
    -- MUST be present if status is "accepted" or "grantedWithMods"
         certificate          REQUIRED
    -- MUST be present when certifiedKeyPair is present
    -- MUST contain the newly enrolled X.509 certificate
       privateKey             OPTIONAL
    -- MUST be absent in case of local key generation or "rejection"
    -- MUST contain the encrypted private key in an EnvelopedData
    --   structure as specified in Section 4.1.6 in case the
    --   private key was generated centrally

protection                    REQUIRED
    -- As described in Section 3.2

extraCerts                    REQUIRED
    -- As described in Section 3.3
    -- MUST contain the chain of the certificate present in
    --   certOrEncCert
    -- Duplicate certificates MAY be omitted


Certificate Confirmation -- certConf

Field                         Value

header
    -- As described in Section 3.1

body
    -- The message of the EE sends a confirmation to the PKI
    --   management entity to accept or reject the issued
    --   certificates
  certConf                    REQUIRED
    -- MUST contain a sequence of one CertStatus
    CertStatus                REQUIRED
      certHash                REQUIRED
    -- The hash algorithm to use MUST be the hash algorithm indicated
    --   in the below hashAlg field.  If the hashAlg field is not
    --   set, it MUST be the hash algorithm defined by the algorithm
    --   identifier of the certificate signature or the dedicated
    --   hash algorithm defined in [RFC9481] for the used certificate
    --   signature algorithm.
      certReqId               REQUIRED
    -- MUST be 0
      statusInfo              OPTIONAL
    -- PKIStatusInfo structure should be present
    -- Omission indicates acceptance of the indicated certificate
       status                 REQUIRED
    -- positive values allowed: "accepted"
    -- negative values allowed: "rejection"
       statusString           OPTIONAL
    -- MAY be any human-readable text for debugging, for logging, or
    --   to display in a GUI
       failInfo               OPTIONAL
    -- MAY be present if status is "rejection"
```

```
    -- MUST be absent if status is "accepted"
      hashAlg                 OPTIONAL
    -- The hash algorithm to use for calculating the above certHash
    -- If used, the pvno field in the header MUST be cmp2021 (3).
    --   For backward compatibility, use of this field is
    --   NOT RECOMMENDED if the hash algorithm to use can be
    --   identified by other means; see above.

protection                  REQUIRED
    -- As described in Section 3.2
    -- MUST use the same credentials as in the first request message
    --   of this PKI management operation

extraCerts                  RECOMMENDED
    -- As described in Section 3.3
    -- MAY be omitted if the message size is critical and the PKI
    --   management entity caches the CMP protection certificate from
    --   the first request message of this PKI management operation


PKI Confirmation -- pkiConf

Field                       Value

header
    -- As described in Section 3.1

body
  pkiconf                   REQUIRED
    -- The content of this field MUST be NULL

protection                  REQUIRED
    -- As described in Section 3.2
    -- MUST use the same credentials as in the first response
    --   message of this PKI management operation

extraCerts                  RECOMMENDED
    -- As described in Section 3.3
    -- MAY be omitted if the message size is critical and the EE has
    --   cached the CMP protection certificate from the first
    --   response message of this PKI management operation
```

4.1.2. Enrolling an End Entity to a Known PKI

This PKI management operation should be used by an EE to request an additional certificate of the same PKI it already has certificates from. The EE uses one of these existing certificates to authenticate itself by signing its request messages using the respective private key.

Specific prerequisites augmenting the prerequisites in Section 3.4 are as follows:

- The certificate used by the EE **MUST** have been enrolled by the PKI it requests another certificate from.
- When using the generalInfo field certProfile, the EE **MUST** know the identifier needed to indicate the requested certificate profile.

The message sequence for this PKI management operation is identical to that given in Section 4.1.1, with the following changes:

1. The body of the first request and response **SHOULD** be cr and cp. Otherwise, ir and ip **MUST** be used.

   Note: Since the difference between ir/ip and cr/cp is syntactically not essential, an ir/ip may be used in this PKI management operation.

2. The caPubs field in the certificate response message **MUST** be absent.

4.1.3. Updating a Valid Certificate

This PKI management operation should be used by an EE to request an update for one of its certificates that is still valid. The EE uses the certificate it wishes to update as the CMP protection certificate. Both for authenticating itself and for proving ownership of the certificate to be updated, it signs the request messages with the corresponding private key.

Specific prerequisites augmenting the prerequisites in Section 3.4 are as follows:

- The certificate the EE wishes to update **MUST NOT** be expired or revoked and **MUST** have been issued by the addressed CA.
- A new public-private key pair should be used.
- When using the generalInfo field certProfile, the EE **MUST** know the identifier needed to indicate the requested certificate profile.

The message sequence for this PKI management operation is identical to that given in Section 4.1.1, with the following changes:

1. The body of the first request and response **MUST** be kur and kup, respectively.
2. Protection of the kur **MUST** be performed using the certificate to be updated.
3. The subject field and/or the subjectAltName extension of the certTemplate **MUST** contain the EE subject name of the existing certificate to be updated, without modifications.
4. The certTemplate **SHOULD** contain the subject and/or subjectAltName extension and publicKey of the EE only.
5. The oldCertId control **MAY** be used to make clear which certificate is to be updated.
6. The caPubs field in the kup message **MUST** be absent.

As part of the certReq structure of the kur, the oldCertId control is added after the certTemplate field.

```
controls
  type                    RECOMMENDED
-- MUST be the value id-regCtrl-oldCertID, if present
  value
    issuer                REQUIRED
    serialNumber          REQUIRED
-- MUST contain the issuer and serialNumber of the certificate
--   to be updated
```

4.1.4. Enrolling an End Entity Using a PKCS #10 Request

This PKI management operation can be used by an EE to request a certificate using the PKCS #10 [RFC2986] format to interoperate with CAs not supporting CRMF [RFC4211]. This offers a variation of the PKI management operations specified in Sections 4.1.1 to 4.1.3.

In this PKI management operation, the public key and all further certificate template data **MUST** be contained in the subjectPKInfo and other certificationRequestInfo fields of the PKCS #10 structure.

The prerequisites are the same as given in Section 4.1.2.

The message sequence for this PKI management operation is identical to that given in Sections 4.1.1 to 4.1.3, with the following changes:

1. The body of the first request and response **MUST** be p10cr and cp, respectively.
2. The certReqId in the cp message **MUST** be -1.

Detailed Message Description:

```
Certification Request -- p10cr

Field                          Value

header
    -- As described in Section 3.1

body
    -- The request of the EE for a new certificate using a PKCS #10
    --   certificate request
  p10cr                        REQUIRED
    certificationRequestInfo   REQUIRED
      version                  REQUIRED
    -- MUST be 0 to indicate PKCS #10 v1.7
      subject                  REQUIRED
    -- The EE subject name MUST be carried in the subject field
    --   and/or the subjectAltName extension.
    -- If subject name is present only in the subjectAltName
    --   extension, then the subject field MUST be NULL-DN
      subjectPKInfo            REQUIRED
        algorithm              REQUIRED
    -- MUST include the subject public key algorithm identifier
        subjectPublicKey       REQUIRED
    -- MUST include the public key to be certified
      attributes               OPTIONAL
    -- MAY include end-entity-specific X.509 extensions of the
    --   requested certificate like subject alternative name,
    --   key usage, and extended key usage
    -- The subjectAltName extension MUST be present if the EE
    --   subject name includes a subject alternative name.
    signatureAlgorithm         REQUIRED
    -- The signature algorithm MUST be consistent with the
    --   subjectPKInfo field.
    signature                  REQUIRED
    -- MUST contain the self-signature for proof-of-possession

protection                     REQUIRED
    -- As described in Section 3.2

extraCerts                     REQUIRED
    -- As described for the underlying PKI management operation
```

4.1.5. Using MAC-Based Protection for Enrollment

This is a variant of the PKI management operations described in Sections 4.1.1, 4.1.2, and 4.1.4. It should be used by an EE to request a certificate of a new PKI in case it does not have a certificate to prove its identity to the target PKI but has some secret information shared with the PKI management entity. Therefore, the request and response messages are MAC-protected using this shared secret information. The distribution of this shared secret is out of scope for this document. The PKI management entity checking the MAC-based protection **MUST** replace this protection according to Section 5.2.3, as the next hop may not know the shared secret information.

Note: The entropy of the shared secret information is crucial for the level of protection when using MAC-based protection. Further guidance is available in the security considerations updated by CMP Updates [RFC9480].

Specific prerequisites augmenting the prerequisites in Section 3.4 are as follows:

- Rather than using private keys, certificates, and trust anchors, the EE and the PKI management entity **MUST** share secret information.

  Note: The shared secret information **MUST** be established out of band, e.g., by a service technician during initial local configuration.

- When using the generalInfo field certProfile, the EE **MUST** know the identifier needed to indicate the requested certificate profile.

The message sequence for this PKI management operation is identical to that given in Sections 4.1.1, 4.1.2, and 4.1.4, with the following changes:

1. The protection of all messages **MUST** be MAC-based. Therefore, extraCerts fields of all messages do not contain CMP protection certificates and associated chains.
2. The sender field **MUST** contain a name the PKI management entity can use to identify the shared secret information used for message protection. This name **MUST** be placed in the commonName field of the directoryName choice. The senderKID **MUST** contain the same name as in the commonName field of the sender field. In case the sending entity does not yet know for which name to request the certificate, it can use this commonName in the subject field of the certTemplate.

See Section 6 of CMP Algorithms [RFC9481] for details on message authentication code algorithms (MSG_MAC_ALG) to use. Typically, parameters are part of the protectionAlg field, e.g., used for key derivation, like a salt and an iteration count. Such parameters should remain constant for message protection throughout this PKI management operation to reduce the computational overhead.

4.1.6. Adding Central Key Pair Generation to Enrollment

This is a variant of the PKI management operations described in Sections 4.1.1 to 4.1.4 and the variant described in Section 4.1.5. It needs to be used in case an EE is not able to generate its new public-private key pair itself or central generation of the EE key material is preferred. Which PKI management entity will act as Key Generation Authority (KGA) and perform the key generation is a matter of the local implementation. This PKI management entity **MUST** use a certificate containing the additional extended key usage extension id-kp-cmKGA in order to be accepted by the EE as a legitimate key generation authority.

Note: As described in Section 5.3.1, the KGA can use the PKI management operation described in Section 4.1.2 to request the certificate for this key pair on behalf of the EE.

When an EE requests central key generation for a certificate update using a kur message, the KGA cannot use a kur message to request the certificate on behalf of the EE, as the old EE credential is not available to the KGA for protecting this message. Therefore, if the EE uses the PKI management operation described in Section 4.1.3, the KGA **MUST** act as described in Section 4.1.2 to request the certificate for the newly generated key pair on behalf of the EE from the CA.

Generally speaking, it is strongly preferable to generate public-private key pairs locally at the EE. This is advisable to make sure that the entity identified in the newly issued certificate is the only entity that knows the private key.

Reasons for central key generation may include the following:

- lack of sufficient initial entropy

  Note: Good random numbers are not only needed for key generation but also for session keys and nonces in any security protocol. Therefore, a decent security architecture should anyways support good random number generation on the EE side or provide enough initial entropy for the random number generator seed to guarantee good pseudorandom number generation. Yet maybe this is not the case at the time of requesting an initial certificate during manufacturing.

- lack of computational resources, in particular, for RSA key generation

  Note: Since key generation could be performed in advance to the certificate enrollment communication, it is often not time critical.

Note: As mentioned in Section 2, central key generation may be required in a push model, where the certificate response message is transferred by the PKI management entity to the EE without a previous request message.

The EE requesting central key generation **MUST** omit the publicKey field from the certTemplate or, in case it has a preference on the key type to be generated, provide this preference in the algorithm sub-field and fill the subjectPublicKey sub-field with a zero-length BIT STRING. Both variants indicate to the PKI management entity that a new key pair shall be generated centrally on behalf of the EE.

Note: As the protection of centrally generated keys in the response message has been extended to EncryptedKey by Section 2.7 of CMP Updates [RFC9480], EnvelopedData is the preferred alternative to EncryptedValue. In CRMF [RFC4211], Section 2.1, point 9, the use of EncryptedValue has been deprecated in favor of the EnvelopedData structure. Therefore, this profile requires using EnvelopedData, as specified in Section 6 of CMS [RFC5652]. When EnvelopedData is to be used in a PKI management operation, CMP v3 **MUST** be indicated in the message header already for the initial request message; see Section 2.20 of CMP Updates [RFC9480].

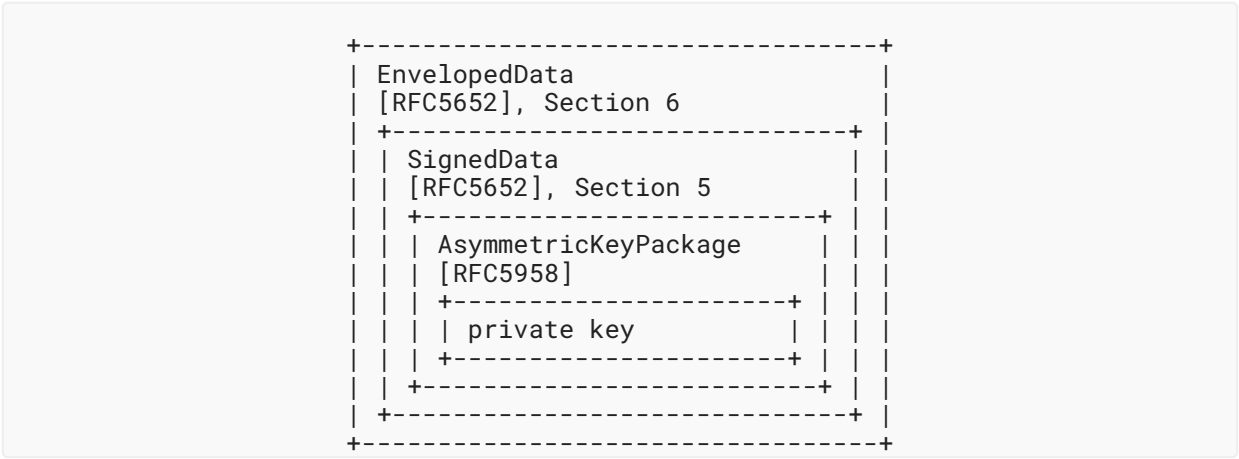


Figure 3: Encrypted Private Key Container

The PKI management entity delivers the private key in the privateKey field in the certifiedKeyPair structure of the response message also containing the newly issued certificate.

The private key **MUST** be provided as an AsymmetricKeyPackage structure as defined in [RFC5958].

This AsymmetricKeyPackage structure **MUST** be wrapped in a SignedData structure, as specified in Section 5 of CMS [RFC5652] and [RFC8933], and signed by the KGA generating the key pair. The signature **MUST** be performed using a private key related to a certificate asserting the extended key usage id-kp-cmKGA, as described in Section 2.2 of CMP Updates [RFC9480], to demonstrate authorization to generate key pairs on behalf of an EE. For response messages using signature-based protection, the EE **MUST** validate the signer certificate contained in the SignedData structure and **SHOULD** authorize the KGA considering any given id-kp-cmKGA extended key usage in the signer certificate. For response messages using MAC-based protection, the EE **MAY** omit the validation as it may not be possible or meaningful to the EE. In this case, the EE authorizes the KGA using the shard secret information.

The SignedData structure **MUST** be wrapped in an EnvelopedData structure, as specified in Section 6 of CMS [RFC5652], encrypting it using a newly generated symmetric content-encryption key.

This content-encryption key **MUST** be securely provided as part of the EnvelopedData structure to the EE using one of three key management techniques. The choice of the key management technique to be used by the PKI management entity depends on the authentication mechanism the EE chose to protect the request message. See Section 2.7 of CMP Updates [RFC9480] for details on which key management technique to use.

- Signature-based protection of the request message:

In this case, the choice depends on the type of public key in the CMP protection certificate used by the EE in its request.

- The content-encryption key **SHALL** be protected using the key transport key management technique (see Section 4.1.6.1) if the key type supports this.
- The content-encryption key **SHALL** be protected using the key agreement key management technique (see Section 4.1.6.2) if the key type supports this.

- MAC-based protection of the request message:
  - The content-encryption key **SHALL** be protected using the password-based key management technique (see Section 4.1.6.3) if and only if the EE used MAC-based protection for the request message.

Specific prerequisites augmenting those of the respective certificate enrollment PKI management operations are as follows:

- If signature-based protection is used, the EE **MUST** be able to authenticate and authorize the KGA using suitable information, which includes a trust anchor.
- If MAC-based protection is used, the KGA **MUST** also know the shared secret information to protect the encrypted transport of the newly generated key pair. Consequently, the EE can also authorize the KGA.
- The PKI management entity **MUST** have a certificate containing the additional extended key usage extension id-kp-cmKGA for signing the SignedData structure containing the private key package.
- For encrypting the SignedData structure, a fresh content-encryption key to be used by the symmetric encryption algorithm **MUST** be generated with sufficient entropy.

  Note: The security strength of the protection of the generated private key should be similar or higher than the security strength of the generated private key.

Detailed Description of the privateKey Field:

```
    privateKey            REQUIRED
-- MUST be an EnvelopedData structure, as specified in
--   Section 6 of CMS [RFC5652]
     version              REQUIRED
-- MUST be 2 for recipientInfo type KeyAgreeRecipientInfo and
--   KeyTransRecipientInfo
-- MUST be 0 for recipientInfo type PasswordRecipientInfo
     recipientInfos       REQUIRED
-- MUST contain a sequence of one RecipientInfo, which MUST be
--   ktri of type KeyTransRecipientInfo (see Section 4.1.6.1),
--   kari of type KeyAgreeRecipientInfo (see Section 4.1.6.2), or
--   pwri of type PasswordRecipientInfo (see Section 4.1.6.3)
     encryptedContentInfo
                          REQUIRED
       contentType        REQUIRED
-- MUST be id-signedData
       contentEncryptionAlgorithm
                          REQUIRED
```

```
    -- MUST be the algorithm identifier of the algorithm used for
    --   content encryption
    -- The algorithm type MUST be PROT_SYM_ALG as specified in
    --   [RFC9481], Section 5
           encryptedContent  REQUIRED
    -- MUST be the SignedData structure, as specified in Section 5
    -- of CMS [RFC5652] and [RFC8933], in encrypted form
            version         REQUIRED
    -- MUST be 3
            digestAlgorithms
                            REQUIRED
    -- MUST contain a sequence of one AlgorithmIdentifier element
    -- MUST be the algorithm identifier of the digest algorithm
    --   used for generating the signature and match the signature
    --   algorithm specified in signatureAlgorithm; see [RFC8933]
            encapContentInfo
                            REQUIRED
    -- MUST contain the content that is to be signed
              eContentType  REQUIRED
    -- MUST be id-ct-KP-aKeyPackage as specified in [RFC5958]
              eContent      REQUIRED
    -- MUST be of type AsymmetricKeyPackage and
    -- MUST contain a sequence of one OneAsymmetricKey element
                version     REQUIRED
    -- MUST be 1 (indicating v2)
                privateKeyAlgorithm
                            REQUIRED
    -- The privateKeyAlgorithm field MUST contain the algorithm
    --   identifier of the asymmetric key pair algorithm
                privateKey  REQUIRED
                publicKey   REQUIRED
    -- MUST contain the public key corresponding to the private key
    --   for simplicity and consistency with v2 of OneAsymmetricKey
            certificates    REQUIRED
    -- MUST contain the certificate for the private key used to sign
    --   the signedData content, together with its chain
    -- The first certificate in this field MUST be the KGA
    --   certificate used for protecting this content
    -- Self-signed certificates should not be included and MUST NOT
    --   be trusted based on their inclusion in any case
              signerInfos   REQUIRED
    -- MUST contain a sequence of one SignerInfo element
              version       REQUIRED
    -- MUST be 3
              sid           REQUIRED
                subjectKeyIdentifier
                            REQUIRED
    -- MUST be the subjectKeyIdentifier of the KGA certificate
              digestAlgorithm
                            REQUIRED
    -- MUST be the same as in the digestAlgorithms field of
    --   encryptedContent
              signedAttrs   REQUIRED
    -- MUST contain an id-contentType attribute containing the value
    --   id-ct-KP-aKeyPackage
    -- MUST contain an id-messageDigest attribute containing the
    --   message digest of eContent
    -- MAY contain an id-signingTime attribute containing the time
```

```
--   of a signature.  It SHOULD be omitted if the transactionTime
--   field is not present in the PKIHeader.
-- For details on the signed attributes, see Sections 5.3 and
--   11 of CMS [RFC5652] and [RFC8933]
           signatureAlgorithm
                          REQUIRED
-- MUST be the algorithm identifier of the signature algorithm
--   used for calculation of the signature bits
-- The signature algorithm type MUST be MSG_SIG_ALG, as
--   specified in [RFC9481], Section 3, and MUST be consistent
--   with the subjectPublicKeyInfo field of the KGA certificate
           signature      REQUIRED
-- MUST be the digital signature of the encapContentInfo
```

As stated in Section 1.8, all fields of the ASN.1 syntax that are defined in [RFC5652] but are not explicitly specified here **SHOULD NOT** be used.

4.1.6.1. Using the Key Transport Key Management Technique

This variant can be applied in combination with the PKI management operations specified in Sections 4.1.1 to 4.1.3 using signature-based protection of CMP messages. The EE certificate used for the signature-based protection of the request message **MUST** contain a public key supporting key transport and allow for the key usage "keyEncipherment". The related key pair **MUST** be used for encipherment of the content-encryption key. For this key management technique, the KeyTransRecipientInfo structure **MUST** be used in the contentInfo field.

The KeyTransRecipientInfo structure included into the EnvelopedData structure is specified in Section 6.2.1 of CMS [RFC5652].

Detailed Description of the KeyTransRecipientInfo Structure:

```
        ktri              REQUIRED
-- MUST be KeyTransRecipientInfo as specified in Section 6.2.1
--   of CMS [RFC5652]
          version         REQUIRED
-- MUST be 2
          rid             REQUIRED
-- MUST contain the subjectKeyIdentifier of the CMP protection
--   certificate, if available, in the rKeyId choice, and the
--   subjectKeyIdentifier MUST equal the senderKID in the
--   PKIHeader.
-- If the CMP protection certificate does not contain a
--   subjectKeyIdentifier, the issuerAndSerialNumber choice MUST
--   be used.
          keyEncryptionAlgorithm
                          REQUIRED
-- MUST be the algorithm identifier of the key transport
--   algorithm.  The algorithm type MUST be KM_KT_ALG as
--   specified in [RFC9481], Section 4.2
          encryptedKey    REQUIRED
-- MUST be the encrypted content-encryption key
```

4.1.6.2. Using the Key Agreement Key Management Technique

This variant can be applied in combination with the PKI management operations specified in Sections 4.1.1 to 4.1.3, using signature-based protection of CMP messages. The EE certificate used for the signature-based protection of the request message **MUST** contain a public key supporting key agreement and allow for the key usage "keyAgreement". The related key pair **MUST** be used for establishment of the content-encryption key. For this key management technique, the KeyAgreeRecipientInfo structure **MUST** be used in the contentInfo field.

The KeyAgreeRecipientInfo structure included into the EnvelopedData structure is specified in Section 6.2.2 of CMS [RFC5652].

Detailed Description of the KeyAgreeRecipientInfo Structure:

```
        kari              REQUIRED
    -- MUST be KeyAgreeRecipientInfo as specified in Section
    --   6.2.2 of CMS [RFC5652]
          version         REQUIRED
    -- MUST be 3
          originator      REQUIRED
    -- MUST contain the subjectKeyIdentifier of the CMP protection
    --   certificate, if available, in the subjectKeyIdentifier
    --   choice, and the subjectKeyIdentifier MUST equal the
    --   senderKID in the PKIHeader.
    -- If the CMP protection certificate does not contain a
    --   subjectKeyIdentifier, the issuerAndSerialNumber choice MUST
    --   be used.
          ukm             RECOMMENDED
    -- MUST be used when 1-Pass Elliptic Curve Menezes-Qu-Vanstone
    --   (ECMQV) is used; see [RFC5753]
    -- SHOULD be present to ensure uniqueness of the key
    --   encryption key
          keyEncryptionAlgorithm
                          REQUIRED
    -- MUST be the algorithm identifier of the key agreement
    --   algorithm
    -- The algorithm type MUST be KM_KA_ALG as specified in
    --   [RFC9481], Section 4.1
    -- The parameters field of the key agreement algorithm MUST
    --   contain the key wrap algorithm.  The algorithm type
    --   MUST be KM_KW_ALG as specified in [RFC9481], Section 4.3
          recipientEncryptedKeys
                          REQUIRED
    -- MUST contain a sequence of one RecipientEncryptedKey
            rid           REQUIRED
    -- MUST contain the subjectKeyIdentifier of the CMP protection
    --   certificate, if available, in the rKeyId choice, and the
    --   subjectKeyIdentifier MUST equal the senderKID in the
    --   PKIHeader.
    -- If the CMP protection certificate does not contain a
    --   subjectKeyIdentifier, the issuerAndSerialNumber choice MUST
    --   be used
              encryptedKey
                          REQUIRED
    -- MUST be the encrypted content-encryption key
```

4.1.6.3. Using the Password-Based Key Management Technique

This variant can be applied in combination with the PKI management operation specified in Section 4.1.5, using MAC-based protection of CMP messages. The shared secret information used for the MAC-based protection **MUST** also be used for the encryption of the content-encryption key but with a different salt value applied in the key derivation algorithm. For this key management technique, the PasswordRecipientInfo structure **MUST** be used in the contentInfo field.

Note: The entropy of the shared secret information is crucial for the level of protection when using a password-based key management technique. For centrally generated key pairs, the entropy of the shared secret information **SHALL NOT** be less than the security strength of the centrally generated key pair. Further guidance is available in Section 9.

The PasswordRecipientInfo structure included into the EnvelopedData structure is specified in Section 6.2.4 of CMS [RFC5652].

Detailed Description of the PasswordRecipientInfo Structure:

```
        pwri              REQUIRED
-- MUST be PasswordRecipientInfo as specified in
--   Section 6.2.4 of CMS [RFC5652]
          version         REQUIRED
-- MUST be 0
          keyDerivationAlgorithm
                          REQUIRED
-- MUST be the algorithm identifier of the key derivation
--   algorithm
-- The algorithm type MUST be KM_KD_ALG as specified in
--   [RFC9481], Section 4.4
          keyEncryptionAlgorithm
                          REQUIRED
-- MUST be the algorithm identifier of the key wrap algorithm
-- The algorithm type MUST be KM_KW_ALG as specified in
--   [RFC9481], Section 4.3
          encryptedKey    REQUIRED
-- MUST be the encrypted content-encryption key
```

4.2. Revoking a Certificate

This PKI management operation should be used by an entity to request revocation of a certificate. Here, the revocation request is used by an EE to revoke one of its own certificates.

The revocation request message **MUST** be signed using the certificate that is to be revoked to prove the authorization to revoke. The revocation request message is signature-protected using this certificate. This requires that the EE still possesses the private key. If this is not the case, the revocation has to be initiated by other means, e.g., revocation by the RA, as specified in Section 5.3.2.

An EE requests revoking a certificate of its own at the CA that issued this certificate. The PKI management entity handles the request as described in Section 5.1.3, and responds with a message that contains the status of the revocation from the CA.

The specific prerequisite augmenting the prerequisites in Section 3.4 is as follows:

- The certificate the EE wishes to revoke is not yet expired or revoked.

Message Flow:

```
Step# EE                                    PKI management entity
 1   format rr
 2                       ->   rr      ->
 3                                        handle or forward rr
 4                                        format or receive rp
 5                       <-   rp      <-
 6   handle rp
```

For this PKI management operation, the EE **MUST** include a sequence of one RevDetails structure in the rr message body. In the case no generic error occurred, the response to the rr **MUST** be an rp message containing a single status field.

Detailed Message Description:

```
Revocation Request -- rr

Field                          Value

header
    -- As described in Section 3.1

body
    -- The request of the EE to revoke its certificate
  rr                           REQUIRED
    -- MUST contain a sequence of one element of type RevDetails
    -- If more revocations are desired, further PKI management
    --   operations need to be initiated
    certDetails                REQUIRED
    -- MUST be present and is of type CertTemplate
      serialNumber             REQUIRED
    -- MUST contain the certificate serialNumber attribute of the
    --   certificate to be revoked
      issuer                   REQUIRED
    -- MUST contain the issuer attribute of the certificate to be
    --   revoked
    crlEntryDetails            REQUIRED
    -- MUST contain a sequence of one reasonCode of type CRLReason
    --   (see [RFC5280], Section 5.3.1)
    -- If the reason for this revocation is not known or shall not
    --   be published, the reasonCode MUST be 0 (unspecified)
protection                     REQUIRED
    -- As described in Section 3.2 and using the private key related
    --   to the certificate to be revoked

extraCerts                     REQUIRED
    -- As described in Section 3.3


Revocation Response -- rp

Field                          Value
```

```
header
    -- As described in Section 3.1

body
    -- The response of the PKI management entity to the request as
    --   appropriate
  rp                          REQUIRED
    status                    REQUIRED
    -- MUST contain a sequence of one element of type PKIStatusInfo
      status                  REQUIRED
    -- positive value allowed: "accepted"
    -- negative value allowed: "rejection"
      statusString            OPTIONAL
    -- MAY be any human-readable text for debugging, for logging, or
    --   to display in a GUI
      failInfo                OPTIONAL
    -- MAY be present if the status is "rejection"
    -- MUST be absent if the status is "accepted"

protection                    REQUIRED
    -- As described in Section 3.2

extraCerts                    REQUIRED
    -- As described in Section 3.3
```

4.3. Support Messages

The following support messages offer on-demand, in-band delivery of content relevant to the EE provided by a PKI management entity. CMP general messages and general response are used for this purpose. Depending on the environment, these requests may be answered by an RA or CA (see also Section 5.1.4).

The general messages and general response messages contain InfoTypeAndValue structures. In addition to those infoType values defined in [RFC4210] and CMP Updates [RFC9480], further OIDs **MAY** be used to define new PKI management operations or new general-purpose support messages as needed in specific environments.

The following contents are specified in this document:

- Get CA certificates.
- Get root CA certificate update.
- Get certificate request template.
- Get new Certificate Revocation Lists (CRLs).

The following message flow and contents are common to all general message (genm) and general response (genp) messages.

Message Flow:

```
Step# EE                                    PKI management entity
 1   format genm
 2                       ->   genm    ->
 3                                       handle or forward genm
 4                                       format or receive genp
 5                       <-   genp    <-
 6   handle genp
```

Detailed Message Description:

```
General Message -- genm

Field                          Value

header
    -- As described in Section 3.1

body
    -- A request by the EE for information
  genm                         REQUIRED
    -- MUST contain a sequence of one element of type
    --   InfoTypeAndValue
    infoType                   REQUIRED
    -- MUST be the OID identifying one of the specific PKI
    --   management operations described below
    infoValue                  OPTIONAL
    -- MUST be as specified for the specific PKI management operation

protection                     REQUIRED
    -- As described in Section 3.2

extraCerts                     REQUIRED
    -- As described in Section 3.3


General Response -- genp

Field                          Value

header
    -- As described in Section 3.1

body
    -- The response of the PKI management entity providing
    --   information
  genp                         REQUIRED
    -- MUST contain a sequence of one element of type
    --   InfoTypeAndValue
    infoType                   REQUIRED
    -- MUST be the OID identifying the specific PKI management
    --   operation described below
    infoValue                  OPTIONAL
    -- MUST be as specified for the specific PKI management operation
```

```
protection                    REQUIRED
    -- As described in Section 3.2

extraCerts                    REQUIRED
    -- As described in Section 3.3
```

4.3.1. Get CA Certificates

This PKI management operation can be used by an EE to request CA certificates from the PKI management entity.

An EE requests CA certificates, e.g., for chain construction, from a PKI management entity by sending a general message with OID id-it-caCerts, as specified in Section 2.14 of CMP Updates [RFC9480]. The PKI management entity responds with a general response with the same OID that either contains a SEQUENCE of certificates populated with the available intermediate and issuing CA certificates or no content in case no CA certificate is available.

No specific prerequisites apply in addition to those specified in Section 3.4.

The message sequence for this PKI management operation is as given above, with the following specific content:

1. the infoType OID to use is id-it-caCerts
2. the infoValue of the request **MUST** be absent
3. if present, the infoValue of the response **MUST** contain a sequence of certificates

Detailed Description of the infoValue Field of genp:

```
      infoValue               OPTIONAL
    -- MUST be absent if no CA certificate is available
    -- MUST be present if CA certificates are available
    --   if present, MUST be a sequence of CMPCertificate
```

4.3.2. Get Root CA Certificate Update

This PKI management operation can be used by an EE to request an updated root CA certificate as described in Section 4.4 of [RFC4210].

An EE requests an update of a root CA certificate from the PKI management entity by sending a general message with OID id-it-rootCaCert. If needed for unique identification, the EE **MUST** include the old root CA certificate in the message body as specified in Section 2.15 of CMP Updates [RFC9480]. The PKI management entity responds with a general response with OID id-it-rootCaKeyUpdate that either contains the update of the root CA certificate consisting of up to three certificates or no content in case no update is available.

Note: This mechanism may also be used to update trusted non-root certificates, e.g., directly trusted intermediate or issuing CA certificates.

The newWithNew certificate is the new root CA certificate and is **REQUIRED** to be present if available. The newWithOld certificate is **REQUIRED** to be present in the response message because it is needed for the receiving entity trusting the old root CA certificate to gain trust in the new root CA certificate. The oldWithNew certificate is **OPTIONAL** because it is only needed in rare scenarios where other entities may not already trust the old root CA.

No specific prerequisites apply in addition to those specified in Section 3.4.

The message sequence for this PKI management operation is as given above, with the following specific content:

1. the infoType OID to use is id-it-rootCaCert in the request and id-it-rootCaKeyUpdate in the response
2. the infoValue of the request **SHOULD** contain the root CA certificate the update is requested for
3. if present, the infoValue of the response **MUST** be a RootCaKeyUpdateContent structure

Detailed Description of the infoValue Field of genm:

```
      infoValue               RECOMMENDED
    -- MUST contain the root CA certificate to be updated if needed
    --   for unique identification
```

Detailed Description of the infoValue Field of genp:

```
      infoValue               OPTIONAL
    -- MUST be absent if no update of the root CA certificate is
    --   available
    -- MUST be present if an update of the root CA certificate
    --   is available and MUST be of type RootCaKeyUpdateContent
        newWithNew            REQUIRED
    -- MUST be present if infoValue is present
    -- MUST contain the new root CA certificate
        newWithOld            REQUIRED
    -- MUST be present if infoValue is present
    -- MUST contain a certificate containing the new public
    --   root CA key signed with the old private root CA key
        oldWithNew            OPTIONAL
    -- MAY be present if infoValue is present
    -- MUST contain a certificate containing the old public
    --   root CA key signed with the new private root CA key
```

4.3.3. Get Certificate Request Template

This PKI management operation can be used by an EE to request a template with parameters for future certificate requests.

An EE requests certificate request parameters from the PKI management entity by sending a general message with OID id-it-certReqTemplate as specified in Section 2.16 of CMP Updates [RFC9480]. The EE **MAY** indicate the certificate profile to use in the id-it-certProfile extension of the generalInfo field in the PKIHeader of the general message as described in Section 3.1. The PKI management entity responds with a general response with the same OID that either contains requirements on the certificate request template or no content in case no specific requirements are imposed by the PKI. The CertReqTemplateValue contains requirements on certificate fields and extensions in a certTemplate. Optionally, it contains a keySpec field containing requirements on algorithms acceptable for key pair generation.

The EE **SHOULD** follow the requirements from the received CertTemplate by including in the certificate requests all the fields requested, taking over all the field values provided and filling in any remaining fields values. The EE **SHOULD NOT** add further fields, name components, and extensions or their (sub)components. If deviating from the recommendations of the template, the certificate request might be rejected.

Note: We deliberately do not use "**MUST**" or "**MUST NOT**" here in order to allow more flexibility in case the rules given here are not sufficient for specific scenarios. The EE can populate the certificate request as wanted and ignore any of the requirements contained in the CertReqTemplateValue. On the other hand, a PKI management entity is free to ignore or replace any parts of the content of the certificate request provided by the EE. The CertReqTemplate PKI management operation offers means to ease a joint understanding of which fields and/or which field values should be used. An example is provided in Appendix A.

In case a field of type Name, e.g., subject, is present in the CertTemplate but has the value NULL-DN (i.e., has an empty list of relative distinguished name (RDN) components), the field **SHOULD** be included in the certificate request and filled with content provided by the EE. Similarly, in case an X.509v3 extension is present but its extnValue is empty, this means that the extension **SHOULD** be included and filled with content provided by the EE. In case a Name component, for instance, a common name or serial number, is given but has an empty string value, the EE **SHOULD** fill in a value. Similarly, in case an extension has subcomponents (e.g., an IP address in a SubjectAltName field) with empty values, the EE **SHOULD** fill in a value.

The EE **MUST** ignore (i.e., not include) empty fields, extensions, and subcomponents that it does not understand or does not know suitable values to fill in.

The publicKey field of type SubjectPublicKeyInfo in the CertTemplate of the CertReqTemplateValue **MUST** be omitted. In case the PKI management entity wishes to make a stipulation on algorithms the EE may use for key generation, this **MUST** be specified using the keySpec field as specified in Section 2.16 of CMP Updates [RFC9480].

The keySpec field, if present, specifies the public key types optionally with parameters and/or RSA key lengths for which a certificate may be requested.

The value of a keySpec element with the OID id-regCtrl-algId, as specified in Section 2.16 of CMP Updates [RFC9480], **MUST** be of type AlgorithmIdentifier and give an algorithm other than RSA. For Elliptic Curve (EC) keys, the curve information **MUST** be specified as described in the respective standard documents.

The value of a keySpec element with the OID id-regCtrl-rsaKeyLen, as specified in Section 2.16 of CMP Updates [RFC9480], **MUST** be a positive integer value and give an RSA key length.

In the CertTemplate of the CertReqTemplateValue, the serialNumber, signingAlg, issuerUID, and subjectUID fields **MUST** be omitted.

The specific prerequisites augmenting the prerequisites in Section 3.4 is as follows:

- When using the generalInfo field certProfile, the EE **MUST** know the identifier needed to indicate the requested certificate profile.

The message sequence for this PKI management operation is as given above, with the following specific content:

1. the infoType OID to use is id-it-certReqTemplate
2. the id-it-certProfile generalInfo field in the header of the request **MAY** contain the name of the requested certificate request template
3. the infoValue of the request **MUST** be absent
4. if present, the infoValue of the response **MUST** be a CertReqTemplateValue containing a CertTemplate structure and an optional keySpec field

Detailed Description of the infoValue Field of genp:

```
      InfoValue               OPTIONAL
    -- MUST be absent if no requirements are available
    -- MUST be present if the PKI management entity has any
    --   requirements on the contents of the certificate template
        certTemplate          REQUIRED
    -- MUST be present if infoValue is present
    -- MUST contain the required CertTemplate structure elements
    -- The SubjectPublicKeyInfo field MUST be absent
        keySpec               OPTIONAL
    -- MUST be absent if no requirements on the public key are
    --   available
    -- MUST be present if the PKI management entity has any
    --   requirements on the keys generated
    -- MUST contain a sequence of one AttributeTypeAndValue per
    --   supported algorithm with attribute id-regCtrl-algId or
    --   id-regCtrl-rsaKeyLen
```

4.3.4. CRL Update Retrieval

This PKI management operation can be used by an EE to request a new CRL. If a CA offers methods to access a CRL, it may include CRL distribution points or authority information access extensions into the issued certificates as specified in [RFC5280]. In addition, CMP offers CRL provisioning functionality as part of the PKI management operation.

An EE requests a CRL update from the PKI management entity by sending a general message with OID id-it-crlStatusList. The EE **MUST** include the CRL source identifying the requested CRL and, if available, the thisUpdate time of the most current CRL instance it already has, as specified in Section 2.17 of CMP Updates [RFC9480]. The PKI management entity **MUST** respond with a general response with OID id-it-crls.

The EE **MUST** identify the requested CRL either by a CRL distribution point name or issuer name.

Note: CRL distribution point names can be obtained from a cRLDistributionPoints extension of a certificate to be validated or from an issuingDistributionPoint extension of the CRL to be updated. CRL issuer names can be obtained from the cRLDistributionPoints extension of a certificate, from the issuer field of the authority key identifier extension of a certificate or CRL, and from the issuer field of a certificate or CRL.

If a thisUpdate value was given, the PKI management entity **MUST** return the latest CRL available from the referenced source if this CRL is more recent than the given thisUpdate time. If no thisUpdate value was given, it **MUST** return the latest CRL available from the referenced source. In all other cases, the infoValue in the response message **MUST** be absent.

The PKI management entity should treat a CRL distribution point name as an internal pointer to identify a CRL that is directly available at the PKI management entity. It is not intended as a way to fetch an arbitrary CRL from an external location, as this location may be unavailable to that PKI management entity.

In addition to the prerequisites specified in Section 3.4, the EE **MUST** know which CRL to request.

Note: If the EE does not want to request a specific CRL, it **MAY** instead use a general message with OID id-it-currentCrl as specified in Section 5.3.19.6 of [RFC4210].

The message sequence for this PKI management operation is as given above, with the following specific content:

1. the infoType OID to use is id-it-crlStatusList in the request and id-it-crls in the response
2. the infoValue of the request **MUST** be present and contain a sequence of one CRLStatus structure
3. if present, the infoValue of the response **MUST** contain a sequence of one CRL

Detailed Description of the infoValue Field of genm:

```
      infoValue               REQUIRED
    -- MUST contain a sequence of one CRLStatus element
       source                 REQUIRED
    -- MUST contain the dpn choice of type DistributionPointName if
    --   the CRL distribution point name is available
    -- Otherwise, MUST contain the issuer choice identifying the CA
    --   that issues the CRL.  It MUST contain the issuer DN in the
    --   directoryName field of a GeneralName element.
       thisUpdate             OPTIONAL
    -- MUST contain the thisUpdate field of the latest CRL the EE
    --   has gotten from the issuer specified in the given dpn or
    --   issuer field
    -- MUST be omitted if the EE does not have any instance of the
    --   requested CRL
```

Detailed Description of the infoValue Field of genp:

```
      infoValue               OPTIONAL
    -- MUST be absent if no CRL to be returned is available
    -- MUST contain a sequence of one CRL update from the referenced
    --   source if a thisUpdate value was not given or a more recent
    --   CRL is available
```

4.4. Handling Delayed Delivery

This is a variant of all PKI management operations described in this document. It is initiated in case a PKI management entity cannot respond to a request message in a timely manner, typically due to offline or asynchronous upstream communication or due to delays in handling the request. The polling mechanism has been specified in Section 5.3.22 of [RFC4210] and updated by [RFC9480].

Depending on the PKI architecture, the entity initiating delayed delivery is not necessarily the PKI management entity directly addressed by the EE.

When initiating delayed delivery of a message received from an EE, the PKI management entity **MUST** respond with a message including the status "waiting". In response to an ir/cr/kur/p10cr message, it must place the status "waiting" in an ip/cp/kup message and for responses to other request message types in an error message. On receiving this response, the EE **MUST** store in its transaction context the senderNonce of the preceding request message because this value will be needed for checking the recipNonce of the final response to be received after polling. It sends a poll request with certReqId 0 if referring to the CertResponse element contained in the ip/cp/kup message, else -1 to refer to the whole message. In case the final response is not yet available, the PKI management entity that initiated the delayed delivery **MUST** answer with a poll response with the same certReqId. The included checkAfter time value indicates the minimum number of seconds that should elapse before the EE sends a new pollReq message to the PKI management

entity. Polling earlier than indicated by the checkAfter value may increase the number of message round trips. This is repeated until a final response is available or any party involved gives up on the current PKI management operation, i.e., a timeout occurs.

When the PKI management entity that initiated delayed delivery can provide the final response for the original request message of the EE, it **MUST** send this response to the EE. Using this response, the EE can continue the current PKI management operation as usual.

No specific prerequisites apply in addition to those of the respective PKI management operation.

Message Flow:

```
Step# EE                                    PKI management entity
 1   format request
       message
 2                  ->     request     ->
 3                                          handle or forward
                                              request
 4                                          format ip/cp/kup/error
                                              with status "waiting"
                                              response in case no
                                              immediate final response
                                              is available
 5                   <- ip/cp/kup/error <-
 6   handle
       ip/cp/kup/error
       with status
       "waiting"

-------------------------  start polling  -------------------------

 7   format pollReq
 8                     ->    pollReq   ->
 9                                          handle or forward pollReq
10                                          in case the final response
                                              for the original request
                                              is available, continue
                                              with step 14
                                            otherwise, format or
                                              receive pollRep with
                                              checkAfter value
11                     <-    pollRep   <-
12   handle pollRep
13   let checkAfter
       time elapse and
       continue with
       step 7

-----------------  end polling, continue as usual  ------------------

14                                          format or receive
                                              final response on
                                              the original request
15                     <-   response   <-
16   handle final
       response
```

Detailed Message Description:

```
Response with Status "waiting" -- ip/cp/kup/error

Field                         Value

header
    -- As described in Section 3.1
```

```
body
    -- As described for the respective PKI management operation, with
    --   the following adaptations:
      status                  REQUIRED -- in case of ip/cp/kup
      pKIStatusInfo           REQUIRED -- in case of error response
    -- PKIStatusInfo structure MUST be present
       status                 REQUIRED
    -- MUST be status "waiting"
       statusString           OPTIONAL
    -- MAY be any human-readable text for debugging, for logging, or
    --   to display in a GUI
       failInfo               PROHIBITED

protection                    REQUIRED
    -- As described in Section 3.2

extraCerts                    OPTIONAL
    -- As described in Section 3.3


Polling Request -- pollReq

Field                         Value

header
    -- As described in Section 3.1

body
    -- The message of the EE asking for the final response or for a
    --   time to check again
  pollReq                     REQUIRED
    certReqId                 REQUIRED
    -- MUST be 0 if referring to a CertResponse element, else -1

protection                    REQUIRED
    -- As described in Section 3.2
    -- MUST use the same credentials as in the first request message
    --   of the PKI management operation

extraCerts                    RECOMMENDED
    -- As described in Section 3.3
    -- MAY be omitted if the message size is critical and the PKI
    --   management entity caches the CMP protection certificate from
    --   the first request message of the PKI management operation


Polling Response -- pollRep

Field                         Value

header
    -- As described in Section 3.1

body
    -- The message indicates the delay after which the EE SHOULD
    --   send another pollReq message for this transaction
  pollRep                     REQUIRED
```

```
    certReqId                 REQUIRED
    -- MUST be 0 if referring to a CertResponse element, else -1
    checkAfter                REQUIRED
    -- MUST be the time in seconds to elapse before a new pollReq
    --   should be sent
    reason                    OPTIONAL
    -- MAY be any human-readable text for debugging, for logging, or
    --   to display in a GUI

protection                    REQUIRED
    -- As described in Section 3.2
    -- MUST use the same credentials as in the first response
    --   message of the PKI management operation

extraCerts                    RECOMMENDED
    -- As described in Section 3.3
    -- MAY be omitted if the message size is critical and the EE has
    --   cached the CMP protection certificate from the first
    --   response message of the PKI management operation


Final Response - Any Type of Response Message

Field                         Value

header
    -- MUST be the header, as described for the response message
    --   of the respective PKI management operation

body
    -- The response of the PKI management entity to the initial
    --   request, as described in the respective PKI management
    --   operation

protection                    REQUIRED
    -- MUST be as described for the response message of the
    --   respective PKI management operation

extraCerts                    REQUIRED
    -- MUST be as described for the response message of the
    --   respective PKI management operation
```

5. PKI Management Entity Operations

This section focuses on request processing by a PKI management entity. Depending on the network and PKI solution design, this can be an RA or CA, any of which may include protocol conversion or central key generation (i.e., acting as a KGA).

A PKI management entity may directly respond to request messages from downstream and report errors. In case the PKI management entity is an RA, it typically forwards the received request messages upstream after checking them and forwards respective response messages downstream. Besides responding to messages or forwarding them, a PKI management entity may

request or revoke certificates on behalf of EEs. A PKI management entity may also need to manage its own certificates and thus act as an EE using the PKI management operations specified in Section 4.

5.1. Responding to Requests

The PKI management entity terminating the PKI management operation at CMP level **MUST** respond to all received requests by returning a related CMP response message or an error. Any intermediate PKI management entity **MAY** respond, depending on the PKI configuration and policy.

In addition to the checks described in Section 3.5, the responding PKI management entity **MUST** check that a request that initiates a new PKI management operation does not use a transactionID that is currently in use. The failInfo bit value to use is transactionIdInUse as described in Section 3.6.4. If any of these verification steps or any of the essential checks described in Section 3.5 and in the following subsections fails, the PKI management entity **MUST** proceed as described in Section 3.6.

The responding PKI management entity **MUST** copy the sender field of the request to the recipient field of the response, **MUST** copy the senderNonce of the request to the recipNonce of the response, and **MUST** use the same transactionID for the response.

5.1.1. Responding to a Certificate Request

An ir/cr/kur/p10cr message is used to request a certificate as described in Section 4.1. The responding PKI management entity **MUST** proceed as follows unless it initiates delayed delivery as described in Section 5.1.5.

The PKI management entity **MUST** check the message body according to the applicable requirements from Section 4.1. Possible failInfo bit values used for error reporting in case a check failed include badCertId and badCertTemplate. It **MUST** verify the presence and value of the proof-of-possession (failInfo bit: badPOP) unless central key generation is requested. If a signature-based proof-of-possession is present, the PKI management entity **MUST** verify, based on local PKI policy, that the subject name in the certTemplate identifies the same entity as the subject name in the CMP protection certificate or matches the identifier used with MAC-based protection. In case this verification fails, the message **MUST** have been protected by an authorized PKI management entity (failInfo bit: notAuthorized). If the special POP value "raVerified" is given, the PKI management entity should check that the request message was signed using a certificate containing the cmcRA extended key usage (failInfo bit: notAuthorized). The PKI management entity should also perform any further checks on the certTemplate contents (failInfo: badCertTemplate) according to any applicable PKI policy and certificate profile.

If the requested certificate is available, the PKI management entity **MUST** respond with a positive ip/cp/kup message as described in Section 4.1.

Note: If central key generation is performed by the responding PKI management entity, the responding PKI management entity **MUST** include the private key in encrypted form in the response as specified in Section 4.1.6.

The prerequisites of the respective PKI management operation specified in Section 4.1 apply.

If the EE requested omission of the certConf message, the PKI management entity **MUST** handle it as described in Section 4.1.1. Therefore, it **MAY** grant this by including the implicitConfirm generalInfo field or including the confirmWaitTime field in the response header.

5.1.2. Responding to a Confirmation Message

A PKI management entity **MUST** handle a certConf message if it has responded before with a positive ip/cp/kup message not granting implicit confirmation. It should check the message body according to the requirements given in Section 4.1.1 (failInfo bit: badCertId) and **MUST** react as described there.

The prerequisites of the respective PKI management operation specified in Section 4.1 apply.

5.1.3. Responding to a Revocation Request

An rr message is used to request revocation of a certificate. The responding PKI management entity should check the message body according to the requirements in Section 4.2. It **MUST** make sure that the referenced certificate exists (failInfo bit: badCertId), has been issued by the addressed CA, and is not already expired or revoked (failInfo bit: certRevoked). On success, it **MUST** respond with a positive rp message, as described in Section 4.2.

No specific prerequisites apply in addition to those specified in Section 3.4.

5.1.4. Responding to a Support Message

A genm message is used to retrieve extra content. The responding PKI management entity should check the message body according to the applicable requirements in Section 4.3 and perform any further checks depending on the PKI policy. On success, it **MUST** respond with a genp message as described there.

Note: The responding PKI management entity may generate the response from scratch or reuse the contents of previous responses. Therefore, it may be worth caching the body of the response message as long as the contained information is valid and current, such that further requests for the same contents can be answered immediately.

No specific prerequisites apply in addition to those specified in Section 3.4.

5.1.5. Initiating Delayed Delivery

This functional extension can be used by a PKI management entity in case the response to a request takes longer than usual. In this case, the PKI management entity should completely validate the request as usual and then start processing the request itself or forward it further upstream as soon as possible. In the meantime, it **MUST** respond with an ip/cp/kup/error message including the status "waiting" and handle subsequent polling as described in Section 4.4.

Typically, as stated in Section 5.2.3, an intermediate PKI management entity should not change the sender and recipient nonces even in case it modifies a request or a response message. In the special case of delayed delivery initiated by an intermediate PKI management entity, there is an

exception. Between the EE and this PKI management entity, pollReq and pollRep messages are exchanged handling the nonces as usual. Yet when the final response from upstream has arrived at the PKI management entity, this response contains the recipNonce copied (as usual) from the senderNonce in the original request message. The PKI management entity that initiated the delayed delivery **MAY** replace the recipNonce in the response message with the senderNonce of the last received pollReq because the downstream entities, including the EE, might expect it in this way. Yet the check specified in Section 3.5 allows alternate use of the senderNonce of the original request.

No specific prerequisites apply in addition to those of the respective PKI management operation.

5.2. Forwarding Messages

In case the PKI solution consists of intermediate PKI management entities (i.e., LRA or RA), each CMP request message coming from an EE or any other downstream PKI management entity **MUST** either be forwarded to the next (upstream) PKI management entity as described in this section, or answered as described in Section 5.1. Any received response message or a locally generated error message **MUST** be forwarded to the next (downstream) PKI entity.

In addition to the checks described in Section 3.5, the forwarding PKI management entity **MAY** verify the proof-of-possession for ir/cr/kur/p10cr messages. If one of these verification procedures fails, the RA proceeds as described in Section 3.6.

A PKI management entity **SHOULD NOT** change the received message unless its role in the PKI system requires it. This is because changes to the message header or body imply reprotection. Changes to the protection breaks end-to-end authentication of the message source. Changes to the certificate template in a certificate request breaks proof-of-possession. More details are available in the following subsections. Concrete PKI system specifications may define when to do so in more detail.

This is particularly relevant in the upstream communication of a request message.

Each forwarding PKI management entity has one or more functionalities. It may:

- verify the identities of EEs and make authorization decisions for certification request processing based on local PKI policy,
- add or modify fields of certificate request messages,
- replace a MAC-based protection with a signature-based protection that can also be verified further upstream and vice versa,
- double-check if the messages transferred back and forth are properly protected and well-formed,
- provide an authentic indication that it has performed all required checks,
- initiate a delayed delivery due to delays transferring messages or handling requests, or
- collect messages from multiple RAs and forward them jointly.

Note: PKI management entities forwarding messages may also store data from a message in a database for later usage or audit purposes. They may also support traversal of a network boundary.

The decision if a message should be forwarded is:

- unchanged with the original protection,
- unchanged with an additional protection, or
- changed with an additional protection

depending on the PKI solution design and the associated security policy, e.g., as defined in the certificate policy (CP) / certification practice statement (CPS) documents [RFC3647].

A PKI management entity **SHOULD** add or **MAY** replace a protection of a message if it

- needs to securely indicate that it has done checks or validations on the message to one of the next (upstream) PKI management entities or
- needs to protect the message using a key and certificate from a different PKI.

If retaining end-to-end message authentication is required, an additional protection **SHALL** be added instead of replacing the original protection.

A PKI management entity **MUST** replace a protection of a message if it

- performs changes to the header or the body of the message or
- needs to convert from or to a MAC-based protection.

This is particularly relevant in the upstream communication of certificate request messages.

Note that the message protection covers only the header and the body and not the extraCerts. The PKI management entity **MAY** change the extraCerts in any of the following message adaptations, e.g., to sort, add, or delete certificates to support subsequent PKI entities. This may be particularly helpful to augment upstream messages with additional certificates or to reduce the number of certificates in downstream messages when forwarding to constrained devices.

5.2.1. Not Changing Protection

This variant means that a PKI management entity forwards a CMP message without changing the header, body, or protection. In this case, the PKI management entity acts more like a proxy, e.g., on a network boundary, implementing no specific RA-like security functionality that requires an authentic indication to the PKI. Still, the PKI management entity might implement checks that result in refusing to forward the request message and instead responding as specified in Section 3.6.

This variant of forwarding a message or the one described in Section 5.2.2.1 **MUST** be used for kur messages and for central key generation.

No specific prerequisites apply in addition to those specified in Section 3.4.

5.2.2. Adding Protection and Batching of Messages

This variant of forwarding a message means that a PKI management entity adds another protection to PKI management messages before forwarding them.

The nested message is a PKI management message containing a PKIMessages sequence as its body, containing one or more CMP messages.

As specified in the updated Section 5.1.3.4 of [RFC4210] (also see Section 2.6 of CMP Updates [RFC9480]), there are various use cases for adding another protection by a PKI management entity. Specific procedures are described in more detail in the following sections.

Detailed Message Description:

```
Nested Message - nested

Field                          Value

header
    -- As described in Section 3.1

body
    -- Container to provide additional protection to original
    --   messages and to bundle request messages or alternatively
    --   response messages
  PKIMessages                  REQUIRED
    -- MUST be a sequence of one or more CMP messages

protection                     REQUIRED
    -- As described in Section 3.2, using the CMP protection key of
    --   the PKI management entity

extraCerts                     REQUIRED
    -- As described in Section 3.3
```

5.2.2.1. Adding Protection to a Request Message

This variant means that a PKI management entity forwards a CMP message while authentically indicating successful validation and approval of a request message without changing the original message authentication.

By adding a protection using its own CMP protection key, the PKI management entity provides a proof of verifying and approving the message, as described above. Thus, the PKI management entity acts as an actual registration authority (RA), which implements important security functionality of the PKI. Applying an additional protection is specifically relevant when forwarding a message that requests a certificate update or central key generation. This is because the original protection of the EE needs to be preserved while adding an indication of approval by the PKI management entity.

The PKI management entity wrapping the original request message in a nested message structure **MUST** copy the values of the senderNonce and transactionID header fields of the original message to the respective header fields of the nested message and apply signature-based protection. The additional signature serves as proof of verification and authorization by this PKI management entity.

The PKI management entity receiving such a nested message that contains a single request message **MUST** validate the additional protection signature on the nested message and check the authorization for the approval it implies. Other fields in the header of the nested message can be ignored.

The PKI management entity responding to the request contained in the nested message sends the response message as described in Section 5.1, without wrapping it in a nested message.

Note: When responding to the inner request message, it must be considered that the verification and approval activity described in this section has already been performed by the PKI management entity that protected the nested message.

Note: This form of nesting messages is characterized by the fact that the transactionID in the header of the nested message is the same as the one used in the included message.

The specific prerequisite augmenting the prerequisites in Section 3.4 is as follows:

- The PKI management entity **MUST** be able to validate the respective request and have the authorization to perform approval of the request according to the PKI policies.

Message Flow:

```
Step# PKI management entity               PKI management entity
 1   format nested
 2                      ->  nested   ->
 3                                        handle or forward nested
 4                                        format or receive response
 5                      <-  response <-
 6   forward response
```

5.2.2.2. Batching Messages

A PKI management entity **MAY** bundle any number of PKI management messages for batch processing or to transfer a bulk of PKI management messages using the nested message structure. In this use case, nested messages are used both on the upstream interface for transferring request messages towards the next PKI management entity and on its downstream interface for response messages.

This PKI management operation is typically used on the interface between an LRA and an RA to bundle several messages for offline or asynchronous delivery. In this case, the LRA needs to initiate delayed delivery, as described in Section 5.1.5. If the RA needs different routing information per the nested PKI management message provided upstream, a suitable mechanism

may need to be implemented to ensure that the downstream delivery of the response is done to the right requester. Since this mechanism strongly depends on the requirements of the target architecture, it is out of scope of this document.

A nested message containing requests is generated locally at the PKI management entity. For the upstream nested message, the PKI management entity acts as a protocol endpoint; therefore, a fresh transactionID and a fresh senderNonce **MUST** be used in the header of the nested message. An upstream nested message may contain request messages, e.g., ir, cr, p10cr, kur, pollReq, certConf, rr, or genm. While building the upstream nested message, the PKI management entity must store the sender, transactionID, and senderNonce fields of all bundled messages together with the transactionID of the upstream nested message.

Such an upstream nested message is sent to the next PKI management entity. The upstream PKI management entity that unbundles it **MUST** handle each of the included request messages as usual. It **MUST** answer with a downstream nested message. This downstream nested message **MUST** use the transactionID of the upstream nested message and return the senderNonce of the upstream nested message as the recipNonce of the downstream nested message. The downstream nested message **MUST** bundle all available individual response messages (e.g., ip, cp, kup, pollRep, pkiConf, rp, genp, or error) for all original request messages of the upstream nested message. While unbundling the downstream nested message, the former PKI management entity must determine lost and unexpected responses based on the previously stored transactionIDs. When it forwards the unbundled responses, any extra messages **MUST** be dropped, and any missing response message **MUST** be answered with an error message (failInfo bit: systemUnavail) to inform the respective requester about the failed certificate management operation.

Note: This form of nesting messages is characterized by the fact that the transactionID in the header of the nested message is different to those used in the included messages.

The protection of the nested messages **MUST NOT** be regarded as an indication of verification or approval of the bundled PKI request messages.

No specific prerequisites apply in addition to those specified in Section 3.4.

Message Flow:

```
Step# PKI management entity               PKI management entity
 1   format nested
 2                      ->  nested   ->
 3                                        handle or forward nested
 4                                        format or receive nested
 5                      <-  nested   <-
 6   handle nested
```

5.2.3. Replacing Protection

The following two alternatives can be used by any PKI management entity forwarding a CMP message with or without changes while providing its own protection and, in this way, asserting approval of the message.

If retaining end-to-end message authentication is required, an additional protection **SHALL** be added instead of replacing the original protection.

By replacing the existing protection using its own CMP protection key, the PKI management entity provides a proof of verifying and approving the message as described above. Thus, the PKI management entity acts as an actual registration authority (RA), which implements important security functionality of the PKI such as verifying the proof of requester identity and authorization.

Note: By replacing the message protection, the binding of a signature-based proof-of-possession to the proof-of-identity given by the original message protection gets lost. To enable the CA to verify this binding, the original message can be provided in the origPKIMessage generalInfo field.

Before replacing the existing protection with a new protection, the PKI management entity:

- **MUST** validate the protection of the received message,
- should check the content of the message,
- may do any modifications that it wants to perform, and
- **MUST** check that the sender of the original message, as authenticated by the message protection, is authorized for the given operation.
- for certificate requests, **MUST** verify the binding of signature-based proof-of-possession to the proof-of-identity as described in Section 5.1.1.

These message adaptations **MUST NOT** be applied to kur messages described in Section 4.1.3 since their original protection using the key and certificate to be updated needs to be preserved.

These message adaptations **MUST NOT** be applied to certificate request messages described in Section 4.1.6 for central key generation since their original protection needs to be preserved up to the KGA, which needs to use it for encrypting the new private key for the EE.

In both the kur and central key generation cases, if a PKI management entity needs to state its approval of the original request message, it **MUST** provide this using a nested message as specified in Section 5.2.2.1.

When an intermediate PKI management entity modifies a message, it **MUST NOT** change the transactionID, the senderNonce, or the recipNonce, apart from the exception for the recipNonce given in Section 5.1.5.

5.2.3.1. Not Changing Proof-of-Possession

This variant of forwarding a message means that a PKI management entity forwards a CMP message with or without modifying the message header or body while preserving any included proof-of-possession.

This variant is typically used when an RA replaces an existing MAC-based protection with its own signature-based protection; because the upstream PKI management entity does not know the respective shared secret information, replacing the protection is useful.

Note: A signature-based proof-of-possession of a certificate request will be broken if any field in the certTemplate structure is changed.

In case the PKI management entity breaks an existing proof-of-possession, the message adaptation described in Section 5.2.3.2 needs to be applied instead.

The specific prerequisite augmenting the prerequisites in Section 3.4 is as follows:

- The PKI management entity **MUST** be able to validate the respective request and have the authorization to perform approval of the request according to the PKI policies.

5.2.3.2. Using raVerified

This variant of forwarding a message needs to be used if a PKI management entity breaks any included proof-of-possession in a certificate request message, for instance, because it forwards an ir or cr message with modifications of the certTemplate, i.e., modification, addition, or removal of fields.

The PKI management entity **MUST** verify the proof-of-possession contained in the original message using the included public key. If successful, the PKI management entity **MUST** change the popo field value to raVerified.

Specific prerequisites augmenting the prerequisites in Section 3.4 are as follows:

- The PKI management entity **MUST** be authorized to replace the proof-of-possession (after verifying it) with raVerified.
- The PKI management entity **MUST** be able to validate the respective request and have the authorization to perform approval of the request according to the PKI policies.

Detailed Description of the popo Field of the certReq Structure:

```
popo
  raVerified              REQUIRED
-- MUST have the value NULL and indicates that the PKI
--   management entity verified the popo of the original message
```

5.3. Acting on Behalf of Other PKI Entities

A PKI management entity may need to request a PKI management operation on behalf of another PKI entity. In this case, the PKI management entity initiates the respective PKI management operation as described in Section 4, acting in the role of the EE.

Note: The request message protection will not authenticate the EE, but it will authenticate the RA acting on behalf of the EE.

5.3.1. Requesting a Certificate

A PKI management entity may use one of the PKI management operations described in Section 4.1 to request a certificate on behalf of another PKI entity. It either generates the key pair itself and inserts the new public key in the subjectPublicKey field of the request certTemplate, or it uses a certificate request received from downstream, e.g., by means of a different protocol. In the latter case, it **MUST** verify the received proof-of-possession if this proof breaks, e.g., due to transformation from PKCS #10 [RFC2986] to CRMF [RFC4211]. It **MUST** also verify, based on local PKI policy, that the subject name in the certTemplate identifies the EE.

No specific prerequisites apply in addition to those specified in Section 4.1.

Note: An upstream PKI management entity will not be able to differentiate this PKI management operation from the one described in Section 5.2.3 because, in both cases, the message is protected by the PKI management entity.

The message sequence for this PKI management operation is identical to the respective PKI management operation given in Section 4.1, with the following changes:

1. The request messages **MUST** be signed using the CMP protection key of the PKI management entity taking the role of the EE in this operation.
2. If inclusion of a proper proof-of-possession is not possible, the PKI management entity **MUST** verify the POP provided from downstream and use "raVerified" in its upstream request.
3. The binding of the proof-of-possession to the proof-of-identity of the requesting EE cannot be provided when acting on behalf of the EE.

5.3.2. Revoking a Certificate

A PKI management entity may use the PKI management operation described in Section 4.2 to revoke a certificate of another PKI entity. This revocation request message **MUST** be signed by the PKI management entity using its own CMP protection key to prove to the PKI authorization to revoke the certificate on behalf of that PKI entity.

No specific prerequisites apply in addition to those specified in Section 4.2.

Note: An upstream PKI management entity will not be able to differentiate this PKI management operation from the ones described in Section 5.2.3.

The message sequence for this PKI management operation is identical to that given in Section 4.2, with the following changes:

1. The rr message **MUST** be signed using the CMP protection key of the PKI management entity acting on behalf of the EE in this operation.

6. CMP Message Transfer Mechanisms

CMP messages are designed to be self-contained, such that, in principle, any reliable transfer mechanism can be used. EEs will typically support only one transfer mechanism. PKI management entities **SHOULD** offer HTTP and **MAY** offer CoAP where required. Piggybacking of CMP messages on any other reliable transfer protocol **MAY** be used, and file-based transfer **MAY** be used in case offline transfer is required.

Independently of the means of transfer, it can happen that messages are lost or that a communication partner does not respond. To prevent waiting indefinitely, each PKI entity that sends CMP requests should use a configurable per-request timeout, and each PKI management entity that handles CMP requests should use a configurable timeout in case a further request message is to be expected from the client side within the same transaction. In this way, a hanging transaction can be closed cleanly with an error as described in Section 3.6 (failInfo bit: systemUnavail), and related resources (for instance, any cached extraCerts) can be freed.

Moreover, there are various situations where the delivery of messages gets delayed. For instance, a serving PKI management entity might take longer than expected to form a response due to administrative processes, resource constraints, or upstream message delivery delays. The transport layer itself may cause delays, for instance, due to offline transport, network segmentation, or intermittent network connectivity. Part of these issues can be detected and handled at CMP level using pollReq and pollRep messages as described in Section 4.4, while others are better handled at transfer level. Depending on the transfer protocol and system architecture, solutions for handling delays at transfer level may be present and can be used for CMP connections, for instance, connection reestablishment and message retransmission.

Note: Long timeout periods are helpful to maximize chances to handle minor delays at lower layers without the need for polling.

Note: When using TCP and similar reliable connection-oriented transport protocols, which is typical in conjunction with HTTP, there is the option to keep the connection alive over multiple request-response message pairs. This may improve efficiency.

When conveying CMP messages in HTTP, CoAP, or MIME-based transfer protocols, the Internet media type "application/pkixcmp" **MUST** be set for transfer encoding as specified in Section 3.4 of CMP over HTTP [RFC6712] and Section 2.3 of CMP over CoAP [RFC9482].

6.1. HTTP Transfer

This transfer mechanism can be used by a PKI entity to transfer CMP messages over HTTP. If HTTP transfer is used, the specifications described in [RFC6712] and updated by CMP Updates [RFC9480] **MUST** be followed.

PKI management operations **MUST** use a URI path consisting of '/.well-known/cmp' or '/.well-known/cmp/p/<name>' as specified in Section 3.3 of CMP Updates [RFC9480]. It **SHOULD** be followed by an operation label depending on the type of PKI management operation.

PKI Management Operation	URI Path Segment	Details
Enrolling an End Entity to a New PKI	initialization	Section 4.1.1
Enrolling an End Entity to a Known PKI	certification	Section 4.1.2
Updating a Valid Certificate	keyupdate	Section 4.1.3
Enrolling an End Entity Using a PKCS #10 Request	pkcs10	Section 4.1.4
Revoking a Certificate	revocation	Section 4.2
Get CA Certificates	getcacerts	Section 4.3.1
Get Root CA Certificate Update	getrootupdate	Section 4.3.2
Get Certificate Request Template	getcertreqtemplate	Section 4.3.3
CRL Update Retrieval	getcrls	Section 4.3.4
Batching Messages Note: This path element is applicable only between PKI management entities.	nested	Section 5.2.2.2

Table 1: HTTP URI Path Segment <operation>

If operation labels are used:

- independently of any variants used (see Sections 4.1.5, 4.1.6, and 4.4), the operation label corresponding to the PKI management operation **SHALL** be used.
- any certConf or pollReq messages **SHALL** be sent to the same endpoint as determined by the PKI management operation.
- when a single request message is nested as described in Section 5.2.2.1, the label to use **SHALL** be the same as for the underlying PKI management operation.

By sending a request to its preferred endpoint, the PKI entity will recognize, via the HTTP response status code, whether a configured URI is supported by the PKI management entity.

In case a PKI management entity receives an unexpected HTTP status code from upstream, it **MUST** respond downstream with an error message as described in Section 3.6, using a failInfo bit corresponding to the status code, e.g., systemFailure.

For certificate management, the major security goal is integrity and data origin authentication. For delivery of centrally generated keys, confidentiality is also a must. These goals are sufficiently achieved by CMP itself, also in an end-to-end fashion.

If a second line of defense is required or general privacy concerns exist, TLS can be used to provide confidentiality on a hop-by-hop basis. TLS should be used with certificate-based authentication to further protect the HTTP transfer as described in [RFC9110]. In addition, the recommendations provided in [RFC9325] should be followed.

Note: The requirements for checking certificates given in [RFC5280] and either [RFC5246] or [RFC8446] must be followed for the TLS layer. Certificate status checking should be used for the TLS certificates of all communication partners.

TLS with mutual authentication based on shared secret information may be used in case no suitable certificates for certificate-based authentication are available, e.g., a PKI management operation with MAC-based protection is used.

Note: The entropy of the shared secret information is crucial for the level of protection available using shard secret information-based TLS authentication. A pre-shared key (PSK) mechanism may be used with shared secret information with an entropy of at least 128 bits. Otherwise, a password-authenticated key exchange (PAKE) protocol is recommended.

Note: The provisioning of client certificates and PSKs is out of scope of this document.

6.2. CoAP Transfer

This transfer mechanism can be used by a PKI entity to transfer CMP messages over CoAP [RFC7252], e.g., in constrained environments. If CoAP transfer is used, the specifications described in CMP over CoAP [RFC9482] **MUST** be followed.

PKI management operations **MUST** use a URI path consisting of '/.well-known/cmp' or '/.well-known/cmp/p/<name>' as specified in Section 2.1 of CMP over CoAP [RFC9482]. It **SHOULD** be followed by an operation label depending on the type of PKI management operation.

PKI Management Operation	URI Path Segment	Details
Enrolling an End Entity to a New PKI	ir	Section 4.1.1
Enrolling an End Entity to a Known PKI	cr	Section 4.1.2

PKI Management Operation	URI Path Segment	Details
Updating a Valid Certificate	kur	Section 4.1.3
Enrolling an End Entity Using a PKCS #10 Request	p10	Section 4.1.4
Revoking a Certificate	rr	Section 4.2
Get CA Certificates	crts	Section 4.3.1
Get Root CA Certificate Update	rcu	Section 4.3.2
Get Certificate Request Template	att	Section 4.3.3
CRL Update Retrieval	crls	Section 4.3.4
Batching Messages Note: This path element is applicable only between PKI management entities.	nest	Section 5.2.2.2

Table 2: CoAP URI Path Segment <operation>

If operation labels are used:

- independently of any variants used (see Sections 4.1.5, 4.1.6, and 4.4), the operation label corresponding to the PKI management operation **SHALL** be used.
- any certConf or pollReq messages **SHALL** be sent to the same endpoint, as determined by the PKI management operation.
- when a single request message is nested as described in Section 5.2.2.1, the label to use **SHALL** be the same as for the underlying PKI management operation.

By sending a request to its preferred endpoint, the PKI entity will recognize, via the CoAP response status code, whether a configured URI is supported by the PKI management entity. The CoAP-inherent discovery mechanisms **MAY** also be used.

In case a PKI management entity receives an unexpected CoAP status code from upstream, it **MUST** respond downstream with an error message, as described in Section 3.6, using a failInfo bit corresponding to the status code, e.g., systemFailure.

Like for HTTP transfer, to offer a second line of defense or to provide hop-by-hop privacy protection, DTLS may be utilized as described in CMP over CoAP [RFC9482]. If DTLS is utilized, the same boundary conditions (peer authentication, etc.) as those stated for TLS to protect HTTP transfer in Section 6.1 apply to DTLS likewise.

Note: The provisioning of client certificates and PSKs is out of scope of this document.

6.3. Piggybacking on Other Reliable Transfer

CMP messages **MAY** also be transferred on some other reliable protocol, e.g., Extensible Authentication Protocol (EAP) or Message Queuing Telemetry Transport (MQTT). Connection, delay, and error handling mechanisms similar to those specified for HTTP in [RFC6712] need to be implemented.

A more detailed specification is out of scope of this document and would need to be given, for instance, in the scope of the transfer protocol used.

6.4. Offline Transfer

For transferring CMP messages between PKI entities, any mechanism that is able to store and forward binary objects of sufficient length and with sufficient reliability while preserving the order of messages for each transaction can be used.

The transfer mechanism should be able to indicate message loss, excessive delay, and possibly other transmission errors. In such cases, the PKI entities **MUST** report an error as specified in Section 3.6, as far as possible.

6.4.1. File-Based Transfer

CMP messages **MAY** be transferred between PKI entities using file-based mechanisms, for instance, when an EE is offline or a PKI management entity performs delayed delivery. Each file **MUST** contain the ASN.1 DER encoding of one CMP message only, where the message may be nested. There **MUST** be no extraneous header or trailer information in the file. The filename extension ".pki" **MUST** be used.

6.4.2. Other Asynchronous Transfer Protocols

Other asynchronous transfer protocols, e.g., email or website upload/download, **MAY** transfer CMP messages between PKI entities. A MIME wrapping is defined for those environments that are MIME-native. The MIME wrapping is specified in Section 3.1 of [RFC8551].

The ASN.1 DER encoding of the CMP messages **MUST** be transferred using the "application/pkixcmp" content type and base64-encoded content transfer encoding, as specified in Section 3.4 of CMP over HTTP [RFC6712]. A filename **MUST** be included either in a "content-type" or a "content-disposition" statement. The filename extension ".pki" **MUST** be used.

7. Conformance Requirements

This section defines which level of support for the various features specified in this profile is required for each type of PKI entity.

7.1. PKI Management Operations

The following table provides an overview of the PKI management operations specified in Sections 4 and 5 and states whether support by conforming EE, RA, and CA implementations is mandatory, recommended, optional, or not applicable. Variants amend or change behavior of base PKI management operations and are therefore also included.

The PKI management operation specifications in Section 4 assume that either the RA or CA is the PKI management entity that terminates the Certificate Management Protocol. If the RA terminates CMP, it either responds directly as described in Section 5.1, or it forwards the certificate management operation towards the CA not using CMP. Section 5.2 describes different options of how an RA can forward a CMP message using CMP. Section 5.3 offers the option that an RA operates on behalf on an EE and therefore takes the role of the EE in Section 4.

ID	PKI Management Operations and Variants	EE	RA	CA
Generic	Generic Aspects of PKI Messages and PKI Management Operations, Section 3	**MUST**	**MUST**	**MUST**
IR	Enrolling an End Entity to a New PKI, Section 4.1.1	**MUST**	**MAY**	**MUST**
CR	Enrolling an End Entity to a Known PKI, Section 4.1.2	**MAY**	**MAY**	**MAY**
KUR	Updating a Valid Certificate, Section 4.1.3	**MUST**	**MAY**	**MUST**
P10CR	Enrolling an End Entity Using a PKCS #10 Request, Section 4.1.4	**MAY**	**MAY**	**MAY**
MAC	Using MAC-Based Protection for Enrollment (IR, CR, and P10CR if supported), Section 4.1.5	**MAY**	**SHOULD** 1)	**MAY**
CKeyGen	Adding Central Key Pair Generation to Enrollment (IR, CR, KUR, and P10CR if supported), Section 4.1.6	**MAY**	**MAY**	**MAY**
RR	Revoking a Certificate, Section 4.2	**SHOULD**	**SHOULD** 2)	**SHOULD** 3)

ID	PKI Management Operations and Variants	EE	RA	CA
CACerts	Get CA Certificates, Section 4.3.1	**MAY**	**MAY**	**MAY**
RootUpd	Get Root CA Certificate Update, Section 4.3.2	**MAY**	**MAY**	**MAY**
ReqTempl	Get Certificate Request Template, Section 4.3.3	**MAY**	**MAY**	**MAY**
CRLUpd	CRL Update Retrieval, Section 4.3.4	**MAY**	**MAY**	**MAY**
Polling	Handling Delayed Delivery, Section 4.4	**MAY**	**MAY**	**MAY**
CertResp	Responding to a Certificate Request (IR, CR, KUR, and P10CR if supported), Section 5.1.1	N/A	**MAY**	**MUST**
CertConf	Responding to a Confirmation Message, Section 5.1.2	N/A	**MAY**	**MUST**
RevResp	Responding to a Revocation Request, Section 5.1.3	N/A	**MAY**	**SHOULD**
GenResp	Responding to a Support Message (CACerts, RootUpd, ReqTempl, CRLUpd if supported), Section 5.1.4	N/A	**MAY**	**MAY**
InitPoll	Initiating Delayed Delivery, Section 5.1.5	N/A	**MAY**	**MAY**
FwdKeep	Forwarding Messages - Not Changing Protection, Section 5.2.1	N/A	**MUST**	N/A
FwdAddS	Forwarding Messages - Adding Protection to a Request Message, Section 5.2.2.1	N/A	**MUST**	**MUST**
FwdAddB	Forwarding Messages - Batching Messages, Section 5.2.2.2	N/A	**MAY**	**MAY**
FwdReqKP	Forwarding Messages - Not Changing Proof-of-Possession, Section 5.2.3.1	N/A	**SHOULD** 1)	N/A
FwdReqBP	Forwarding Messages - Using raVerified, Section 5.2.3.2	N/A	**MAY**	**MAY**
CertROnB	Acting on Behalf of Other PKI Entities - Requesting a Certificate, Section 5.3.1	N/A	**MAY**	N/A

ID	PKI Management Operations and Variants	EE	RA	CA
RevROnB	Acting on Behalf of Other PKI Entities - Revoking a Certificate, Section 5.3.2	N/A	**SHOULD** 2)	**SHOULD** 3)

Table 3: Level of Support for PKI Management Operations and Variants

1) The RA should be able to change the CMP message protection from MAC-based to signature-based protection; see Section 5.2.3.1.
2) The RA should be able to request certificate revocation on behalf of an EE (see Section 5.3.2), e.g., in order to handle incidents.
3) An alternative would be to perform revocation at the CA without using CMP, for instance, using a local administration interface.

7.2. Message Transfer

CMP does not have specific needs regarding message transfer, except that, for each request message sent, eventually a sequence of one response message should be received. Therefore, virtually any reliable transfer mechanism can be used, such as HTTP, CoAP, and file-based offline transfer. Thus, this document does not require any specific transfer protocol to be supported by conforming implementations.

On different links between PKI entities (e.g., EE-RA and RA-CA), different transfer mechanisms, as specified in Section 6, may be used.

HTTP **SHOULD** be supported and CoAP **MAY** be supported at all PKI entities for maximizing general interoperability at transfer level. Yet full flexibility is retained to choose whatever transfer mechanism is suitable, for instance, for devices and system architectures with specific constraints.

The following table lists the name and level of support specified for each transfer mechanism.

ID	Message Transfer Type	EE	RA	CA
HTTP	HTTP Transfer, Section 6.1	**SHOULD**	**SHOULD**	**SHOULD**
CoAP	CoAP Transfer, Section 6.2	**MAY**	**MAY**	**MAY**
Piggyb	Piggybacking on Other Reliable Transfer, Section 6.3	**MAY**	**MAY**	**MAY**
Offline	Offline Transfer, Section 6.4	**MAY**	**MAY**	**MAY**

Table 4: Level of Support for Message Transfer Types

8. IANA Considerations

IANA has registered the following content in the "CMP Well-Known URI Path Segments" registry (see <https://www.iana.org/assignments/cmp>), as defined in [RFC8615].

Path Segment	Description	Reference
initialization	Enrolling an End Entity to a New PKI over HTTP	RFC 9483, Section 4.1.1
certification	Enrolling an End Entity to a Known PKI over HTTP	RFC 9483, Section 4.1.2
keyupdate	Updating a Valid Certificate over HTTP	RFC 9483, Section 4.1.3
pkcs10	Enrolling an End Entity Using a PKCS #10 Request over HTTP	RFC 9483, Section 4.1.4
revocation	Revoking a Certificate over HTTP	RFC 9483, Section 4.2
getcacerts	Get CA Certificates over HTTP	RFC 9483, Section 4.3.1
getrootupdate	Get Root CA Certificate Update over HTTP	RFC 9483, Section 4.3.2
getcertreqtemplate	Get Certificate Request Template over HTTP	RFC 9483, Section 4.3.3
getcrls	CRL Update Retrieval over HTTP	RFC 9483, Section 4.3.4
nested	Batching Messages over HTTP	RFC 9483, Section 5.2.2.2
ir	Enrolling an End Entity to a New PKI over CoAP	RFC 9483, Section 4.1.1
cr	Enrolling an End Entity to a Known PKI over CoAP	RFC 9483, Section 4.1.2
kur	Updating a Valid Certificate over CoAP	RFC 9483, Section 4.1.3

Path Segment	Description	Reference
p10	Enrolling an End Entity Using a PKCS #10 Request over CoAP	RFC 9483, Section 4.1.4
rr	Revoking a Certificate over CoAP	RFC 9483, Section 4.2
crts	Get CA Certificates over CoAP	RFC 9483, Section 4.3.1
rcu	Get Root CA Certificate Update over CoAP	RFC 9483, Section 4.3.2
att	Get Certificate Request Template over CoAP	RFC 9483, Section 4.3.3
crls	CRL Update Retrieval over CoAP	RFC 9483, Section 4.3.4
nest	Batching Messages over CoAP	RFC 9483, Section 5.2.2.2

Table 5: New "CMP Well-Known URI Path Segments" Registry Entries

9. Security Considerations

The security considerations laid out in CMP [RFC4210] and updated by CMP Updates [RFC9480], CMP Algorithms [RFC9481], CRMF [RFC4211], Algorithm Requirements Update [RFC9045], CMP over HTTP [RFC6712], and CMP over CoAP [RFC9482] apply.

Trust anchors for chain validations are often provided in the form of self-signed certificates. All trust anchors **MUST** be stored on the device with integrity protection. In some cases, a PKI entity may not have sufficient storage for the complete certificates. In such cases, it may only store, e.g., a hash of each self-signed certificate and require receiving the certificate in the extraCerts field, as described in Section 3.3. If such self-signed certificates are provided in-band in the messages, they **MUST** be verified using information from the trust store of the PKI entity.

For TLS using shared secret information-based authentication, both PSK and PAKE provide the same amount of protection against a real-time authentication attack, which is directly the amount of entropy in the shared secret. The difference between a pre-shared key (PSK) and a password-authenticated key exchange (PAKE) protocol is in the level of long-term confidentiality of the TLS messages against brute-force decryption, where a PSK-based cipher suite only provides security according to the entropy of the shared secret, while a PAKE-based cipher suite provides full security independent of the entropy of the shared secret.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <https://www.rfc-editor.org/info/rfc4210>.

[RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, DOI 10.17487/RFC4211, September 2005, <https://www.rfc-editor.org/info/rfc4211>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <https://www.rfc-editor.org/info/rfc5958>.

[RFC6712] Kause, T. and M. Peylo, "Internet X.509 Public Key Infrastructure -- HTTP Transfer for the Certificate Management Protocol (CMP)", RFC 6712, DOI 10.17487/RFC6712, September 2012, <https://www.rfc-editor.org/info/rfc6712>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

[RFC8933] Housley, R., "Update to the Cryptographic Message Syntax (CMS) for Algorithm Identifier Protection", RFC 8933, DOI 10.17487/RFC8933, October 2020, <https://www.rfc-editor.org/info/rfc8933>.

[RFC9045] Housley, R., "Algorithm Requirements Update to the Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 9045, DOI 10.17487/RFC9045, June 2021, <https://www.rfc-editor.org/info/rfc9045>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC9325] Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <https://www.rfc-editor.org/info/rfc9325>.

[RFC9480] Brockhaus, H., von Oheimb, D., and J. Gray, "Certificate Management Protocol (CMP) Updates", RFC 9480, DOI 10.17487/RFC9480, November 2023, <https://www.rfc-editor.org/info/rfc9480>.

[RFC9481] Brockhaus, H., Aschauer, H., Ounsworth, M., and J. Gray, "Certificate Management Protocol (CMP) Algorithms", RFC 9481, DOI 10.17487/RFC9481, November 2023, <https://www.rfc-editor.org/info/rfc9481>.

[RFC9482] Sahni, M., Ed. and S. Tripathi, Ed., "Constrained Application Protocol (CoAP) Transfer for the Certificate Management Protocol", RFC 9482, DOI 10.17487/RFC9482, November 2023, <https://www.rfc-editor.org/info/rfc9482>.

10.2. Informative References

[BRSKI-AE] von Oheimb, D., Fries, S., and H. Brockhaus, "BRSKI-AE: Alternative Enrollment Protocols in BRSKI", Work in Progress, Internet-Draft, draft-ietf-anima-brski-ae-05, 28 June 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-anima-brski-ae-05>.

[BRSKI-PRM] Fries, S., Werner, T., Lear, E., and M. Richardson, "BRSKI with Pledge in Responder Mode (BRSKI-PRM)", Work in Progress, Internet-Draft, draft-ietf-anima-brski-prm-10, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-anima-brski-prm-10>.

[ETSI-3GPP.33.310] 3GPP, "Network Domain Security (NDS); Authentication Framework (AF)", 3GPP TS 33.310 16.6.0, December 2020, <http://www.3gpp.org/ftp/Specs/html-info/33310.htm>.

[ETSI-EN.319411-1] ETSI, "Electronic Signatures and Infrastructures (ESI); Policy and security requirements for Trust Service Providers issuing certificates; Part 1: General requirements", V1.3.1, ETSI EN 319 411-1, May 2021, <https://www.etsi.org/deliver/etsi_en/319400_319499/31941101/01.03.01_60/en_31941101v010301p.pdf>.

[HTTP-CMP] Brockhaus, H., von Oheimb, D., Ounsworth, M., and J. Gray, "Internet X.509 Public Key Infrastructure -- HTTP Transfer for the Certificate Management Protocol (CMP)", Work in Progress, Internet-Draft, draft-ietf-lamps-rfc6712bis-03, 10 February 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-rfc6712bis-03>.

[IEC.62443-3-3] IEC, "Industrial communication networks - Network and system security - Part 3-3: System security requirements and security levels", IEC 62443-3-3:2013, August 2013, <https://webstore.iec.ch/publication/7033>.

[IEEE.802.1AR_2018] IEEE, "IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity", IEEE Std 802.1AR-2018, DOI 10.1109/IEEESTD.2018.8423794, August 2018, <https://ieeexplore.ieee.org/document/8423794>.

[NIST.CSWP.04162018] National Institute of Standards and Technology (NIST), "Framework for Improving Critical Infrastructure Cybersecurity", Version 1.1, DOI 10.6028/NIST.CSWP.04162018, April 2018, <http://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf>.

[NIST.SP.800-57p1r5] Barker, E., "Recommendation for Key Management: Part 1 - General", DOI 10.6028/NIST.SP.800-57pt1r5, May 2020, <https://doi.org/10.6028/NIST.SP.800-57pt1r5>.

[PKIX-CMP] Brockhaus, H., von Oheimb, D., Ounsworth, M., and J. Gray, "Internet X.509 Public Key Infrastructure -- Certificate Management Protocol (CMP)", Work in Progress, Internet-Draft, draft-ietf-lamps-rfc4210bis-07, 19 June 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-rfc4210bis-07>.

[RFC3647] Chokhani, S., Ford, W., Sabett, R., Merrill, C., and S. Wu, "Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework", RFC 3647, DOI 10.17487/RFC3647, November 2003, <https://www.rfc-editor.org/info/rfc3647>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC5753] Turner, S. and D. Brown, "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)", RFC 5753, DOI 10.17487/RFC5753, January 2010, <https://www.rfc-editor.org/info/rfc5753>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <https://www.rfc-editor.org/info/rfc7030>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <https://www.rfc-editor.org/info/rfc8366>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

[RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <https://www.rfc-editor.org/info/rfc8572>.

[RFC8649] Housley, R., "Hash Of Root Key Certificate Extension", RFC 8649, DOI 10.17487/RFC8649, August 2019, <https://www.rfc-editor.org/info/rfc8649>.

[RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <https://www.rfc-editor.org/info/rfc8995>.

[SZTP-CSR] Watsen, K., Housley, R., and S. Turner, "Conveying a Certificate Signing Request (CSR) in a Secure Zero Touch Provisioning (SZTP) Bootstrapping Request", Work in Progress, Internet-Draft, draft-ietf-netconf-sztp-csr-14, 2 March 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-sztp-csr-14>.

[UNISIG.Subset-137] UNISIG, "ERTMS/ETCS On-line Key Management FFFIS", Subset-137, V1.0.0, December 2015, <https://www.era.europa.eu/system/files/2023-01/sos3_index083_-_subset-137_v100.pdf>.

Appendix A. Example CertReqTemplate

Suppose the server requires that the certTemplate contains:

- the issuer field with a value to be filled in by the EE,
- the subject field with a common name to be filled in by the EE and two organizational unit fields with given values "myDept" and "myGroup",
- the publicKey field contains an Elliptic Curve Cryptography (ECC) key on curve secp256r1 or an RSA public key of length 2048,
- the subjectAltName extension with DNS name "www.myServer.com" and an IP address to be filled in,
- the keyUsage extension marked critical with the value digitalSignature and keyAgreement, and
- the extKeyUsage extension with values to be filled in by the EE.

Then the infoValue with certTemplate and keySpec fields returned to the EE will be encoded as follows:

```
SEQUENCE {
  SEQUENCE {
    [3] {
      SEQUENCE {}
      }
    [5] {
      SEQUENCE {
        SET {
          SEQUENCE {
            OBJECT IDENTIFIER commonName (2 5 4 3)
            UTF8String ""
            }
          }
        SET {
          SEQUENCE {
            OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
            UTF8String "myDept"
            }
          }
        SET {
          SEQUENCE {
            OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
            UTF8String "myGroup"
            }
          }
        }
      }
    [9] {
      SEQUENCE {
        OBJECT IDENTIFIER subjectAltName (2 5 29 17)
        OCTET STRING, encapsulates {
          SEQUENCE {
            [2] "www.myServer.com"
            [7] ""
            }
          }
        }
      SEQUENCE {
        OBJECT IDENTIFIER keyUsage (2 5 29 15)
        BOOLEAN TRUE
        OCTET STRING, encapsulates {
          BIT STRING 3 unused bits
            "10001"B
          }
        }
      SEQUENCE {
        OBJECT IDENTIFIER extKeyUsage (2 5 29 37)
        OCTET STRING, encapsulates {
          SEQUENCE {}
          }
        }
      }
    }
  SEQUENCE {
    SEQUENCE {
      OBJECT IDENTIFIER algId (1 3 6 1 5 5 7 5 1 11)
        SEQUENCE {
```

```
          OBJECT IDENTIFIER ecPublicKey (1 2 840 10045 2 1)
          OBJECT IDENTIFIER secp256r1 (1 2 840 10045 3 1 7)
          }
      }
    SEQUENCE {
      OBJECT IDENTIFIER rsaKeyLen (1 3 6 1 5 5 7 5 1 12)
      INTEGER 2048
      }
    }
  }
```

Acknowledgements

We thank the various reviewers of this document.

Authors' Addresses

Hendrik Brockhaus
Siemens
Werner-von-Siemens-Strasse 1
80333 Munich
Germany
Email: hendrik.brockhaus@siemens.com
URI: https://www.siemens.com

David von Oheimb
Siemens
Werner-von-Siemens-Strasse 1
80333 Munich
Germany
Email: david.von.oheimb@siemens.com
URI: https://www.siemens.com

Steffen Fries
Siemens AG
Werner-von-Siemens-Strasse 1
80333 Munich
Germany
Email: steffen.fries@siemens.com
URI: https://www.siemens.com