Stream:	Internet Engineering Task Force (IETF)
RFC:	9480
Updates:	4210, 5912, 6712
Category:	Standards Track
Published:	October 2023
ISSN:	2070-1721
Authors:	H. Brockhaus	D. von Oheimb	J. Gray
---	---	---	---
	Siemens	*Siemens*	*Entrust*

RFC 9480
Certificate Management Protocol (CMP) Updates

Abstract

This document contains a set of updates to the syntax of Certificate Management Protocol (CMP) version 2 and its HTTP transfer mechanism. This document updates RFCs 4210, 5912, and 6712.

The aspects of CMP updated in this document are using EnvelopedData instead of EncryptedValue, clarifying the handling of p10cr messages, improving the crypto agility, as well as adding new general message types, extended key usages to identify certificates for use with CMP, and well-known URI path segments.

CMP version 3 is introduced to enable signaling support of EnvelopedData instead of EncryptedValue and signal the use of an explicit hash AlgorithmIdentifier in certConf messages, as far as needed.

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/rfc9480.

Copyright Notice

Table of Contents

1. Introduction

While using CMP [RFC4210] in industrial and Internet of Things environments and developing the Lightweight CMP Profile [RFC9483], some limitations were identified in the original CMP specification. This document updates [RFC4210] and [RFC6712] to overcome these limitations.

Among other updates, this document improves the crypto agility of CMP, which allows more flexibility for future advances in cryptography.

This document also introduces new extended key usages to identify CMP endpoints on registration and certification authorities.

The main content of [RFC4210] and [RFC6712] remains unchanged. This document lists all sections that are updated, replaced, or added to the current text of the respective RFCs.

The authors acknowledge that the style of the document is hard to read because the original RFCs must be read along with this document to get the complete content. The working group decided to use this approach in order to keep the changes to [RFC4210] and [RFC6712] to the required minimum. This was meant to speed up the editorial process and to minimize the effort spent on reviewing the full text of the original documents.

However, [PKIX-CMP] and [HTTP-CMP] are intended to obsolete RFCs 4210 and 6712, respectively; these documents also include the changes listed in this document.

1.1. Convention 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.

Technical terminology is used in conformance with [RFC4210], [RFC4211], and [RFC5280]. The following key words are used:

CA: Certification authority, which issues certificates.

RA: Registration authority, an optional system component to which a CA delegates certificate management functions, such as authorization checks.

KGA: Key generation authority, which generates key pairs on behalf of an EE. The KGA could be colocated with an RA or a CA.

EE: End entity, a user, device, or service that holds a PKI certificate. An identifier for the EE is given as its subject of the certificate.

2. Updates to RFC 4210 - Certificate Management Protocol (CMP)

2.1. New Section 1.1 - Changes Since RFC 4210

The following subsection describes feature updates to [RFC4210]. They are always related to the base specification. Hence, references to the original sections in [RFC4210] are used whenever possible.

Insert this section after the current Section 1 of [RFC4210]:

1.1. Changes Since RFC 4210

The following updates are made in this document:

- Adding new extended key usages for various CMP server types, e.g., registration authority and certification authority, to express the authorization of the entity identified in the certificate containing the respective extended key usage extension that acts as the indicated PKI management entity.
- Extending the description of multiple protection to cover additional use cases, e.g., batch processing of messages.
- Offering EnvelopedData as the preferred choice next to EncryptedValue to better support crypto agility in CMP. Note that, according to [RFC4211], Section 2.1, point 9, the use of the EncryptedValue structure has been deprecated in favor of the EnvelopedData structure. [RFC4211] offers the EncryptedKey structure a choice of EncryptedValue and EnvelopedData for migration to EnvelopedData. For reasons of completeness and consistency, the type EncryptedValue has been exchanged in all occurrences in [RFC4210]. This includes the protection of centrally generated private keys, encryption of certificates, and protection of revocation passphrases. To properly differentiate the support of EnvelopedData instead of EncryptedValue, CMP version 3 is introduced in case a transaction is supposed to use EnvelopedData.
- Offering an optional hashAlg field in CertStatus that supports confirmation of certificates signed with signature algorithms, e.g., preparing for upcoming post quantum algorithms, not directly indicating a specific hash algorithm to use to compute the certHash.
- Adding new general message types to request CA certificates, a root CA update, a certificate request template, or a Certificate Revocation List (CRL) update.
- Extending the usage of polling to p10cr, certConf, rr, genm, and error messages.
- Deleting the mandatory algorithm profile in Appendix D.2 of [RFC4210] and referring to Section 7 of CMP Algorithms [RFC9481].

2.2. New Section 4.5 - Extended Key Usage

The following subsection introduces a new extended key usage for CMP servers authorized to centrally generate key pairs on behalf of end entities.

Insert this section after Section 4.4.3 of [RFC4210]:

4.5. Extended Key Usage

The extended key usage (EKU) extension indicates the purposes for which the certified key pair may be used. Therefore, it restricts the use of a certificate to specific applications.

A CA may want to delegate parts of its duties to other PKI management entities. This section provides a mechanism to both prove this delegation and enable an automated means for checking the authorization of this delegation. Such delegation may also be expressed by other means, e.g., explicit configuration.

To offer automatic validation for the delegation of a role by a CA to another entity, the certificates used for CMP message protection or signed data for central key generation **MUST** be issued by the delegating CA and **MUST** contain the respective EKUs. This proves the authorization of this entity by delegating CA to act in the given role, as described below.

The OIDs to be used for these EKUs are:

```
id-kp-cmcCA OBJECT IDENTIFIER ::= {
   iso(1) identified-organization(3) dod(6) internet(1)
   security(5) mechanisms(5) pkix(7) kp(3) 27 }

id-kp-cmcRA OBJECT IDENTIFIER ::= {
   iso(1) identified-organization(3) dod(6) internet(1)
   security(5) mechanisms(5) pkix(7) kp(3) 28 }

id-kp-cmKGA OBJECT IDENTIFIER ::= {
   iso(1) identified-organization(3) dod(6) internet(1)
   security(5) mechanisms(5) pkix(7) kp(3) 32 }
```

Note: Section 2.10 of [RFC6402] specifies OIDs for a Certificate Management over CMS (CMC) CA and a CMC RA. As the functionality of a CA and RA is not specific to any certificate management protocol (such as CMC or CMP), these EKUs are reused by CMP.

The meaning of the id-kp-cmKGA EKU is as follows:

CMP KGA: CMP key generation authorities are CAs or are identified by the id-kp-cmKGA extended key usage. The CMP KGA knows the private key it generated on behalf of the end entity. This is a very sensitive service and needs specific authorization, which by default is with the CA certificate itself. The CA may delegate its authorization by

placing the id-kp-cmKGA extended key usage in the certificate used to authenticate the origin of the generated private key. The authorization may also be determined through local configuration of the end entity.

2.3. Update Section 5.1.1 - PKI Message Header

Section 5.1.1 of [RFC4210] describes the PKI message header. This document introduces the new version 3, indicating support of EnvelopedData as specified in Section 2.7 and hashAlg as specified in Section 2.10.

Replace the ASN.1 syntax of PKIHeader and the subsequent description of pvno with the following text:

```
PKIHeader ::= SEQUENCE {
   pvno                INTEGER     { cmp1999(1), cmp2000(2),
                                     cmp2021(3) },
   sender              GeneralName,
   recipient           GeneralName,
   messageTime     [0] GeneralizedTime         OPTIONAL,
   protectionAlg   [1] AlgorithmIdentifier{ALGORITHM, {...}}
                       OPTIONAL,
   senderKID       [2] KeyIdentifier           OPTIONAL,
   recipKID        [3] KeyIdentifier           OPTIONAL,
   transactionID   [4] OCTET STRING            OPTIONAL,
   senderNonce     [5] OCTET STRING            OPTIONAL,
   recipNonce      [6] OCTET STRING            OPTIONAL,
   freeText        [7] PKIFreeText             OPTIONAL,
   generalInfo     [8] SEQUENCE SIZE (1..MAX) OF
                       InfoTypeAndValue     OPTIONAL
}

PKIFreeText ::= SEQUENCE SIZE (1..MAX) OF UTF8String
```

The usage of the protocol version number (pvno) is described in Section 7.

2.4. New Section 5.1.1.3 - CertProfile

Section 5.1.1 of [RFC4210] defines the PKIHeader and id-it OIDs to be used in the generalInfo field. This section introduces id-it-certProfile.

Insert this section after Section 5.1.1.2 of [RFC4210]:

5.1.1.3. CertProfile

This is used by the EE to indicate specific certificate profiles, e.g., when requesting a new certificate or a certificate request template; see Section 5.3.19.16.

```
id-it-certProfile  OBJECT IDENTIFIER ::= {id-it 21}
CertProfileValue ::= SEQUENCE SIZE (1..MAX) OF UTF8String
```

When used in an ir/cr/kur/genm, the value **MUST NOT** contain more elements than the number of CertReqMsg or InfoTypeAndValue elements and the certificate profile names refer to the elements in the given order.

When used in a p10cr, the value **MUST NOT** contain multiple certificate profile names.

2.5. Update Section 5.1.3.1 - Shared Secret Information

Section 5.1.3.1 of [RFC4210] describes the protection of a PKIMessage based on message authentication code (MAC) using the algorithm id-PasswordBasedMac.

Replace the first paragraph with the following text:

In this case, the sender and recipient share secret information with sufficient entropy (established via out-of-band means or from a previous PKI management operation). PKIProtection will contain a MAC value and the protectionAlg **MAY** be one of the options described in CMP Algorithms [RFC9481]. The PasswordBasedMac is specified as follows (see also [RFC4211] and [RFC9045]):

Replace the last paragraph with the following text (Note: This fixes Errata ID 2616):

Note: It is **RECOMMENDED** that the fields of PBMParameter remain constant throughout the messages of a single transaction (e.g., ir/ip/certConf/pkiConf) to reduce the overhead associated with PasswordBasedMac computation.

2.6. Replace Section 5.1.3.4 - Multiple Protection

Section 5.1.3.4 of [RFC4210] describes the nested message. This document also enables using nested messages for batch-delivery transport of PKI messages between PKI management entities and with mixed body types.

Replace the text of the section with the following text:

5.1.3.4. Multiple Protection

When receiving a protected PKI message, a PKI management entity, such as an RA, **MAY** forward that message along with adding its own protection (which is a MAC or a signature, depending on the information and certificates shared between the RA and the CA). Additionally, multiple PKI messages **MAY** be aggregated. There are several use cases for such messages.

- The RA confirms having validated and authorized a message and forwards the original message unchanged.
- The RA modifies the message(s) in some way (e.g., adds or modifies particular field values or adds new extensions) before forwarding them; then, it **MAY** create its own desired PKIBody. If the changes made by the RA to PKIMessage break the POP of a certificate request, the RA **MUST** set the popo field to RAVerified. It **MAY** include the original PKIMessage from the EE in the generalInfo field of PKIHeader of a nested message (to accommodate, for example, cases in which the CA wishes to check POP or other information on the original EE message). The

infoType to be used in this situation is {id-it 15} (see Section 5.3.19 for the value of id-it), and the infoValue is PKIMessages (contents **MUST** be in the same order as the message in PKIBody).

- A PKI management entity collects several messages that are to be forwarded in the same direction and forwards them in a batch. Request messages can be transferred as batch upstream (towards the CA); response or announce messages can be transferred as batch downstream (towards an RA but not to the EE). For instance, this can be used when bridging an off-line connection between two PKI management entities.

These use cases are accomplished by nesting the messages within a new PKI message. The structure used is as follows:

```
NestedMessageContent ::= PKIMessages
```

2.7. Replace Section 5.2.2 - Encrypted Values

Section 5.2.2 of [RFC4210] describes the use of EncryptedValue to transport encrypted data. This document extends the encryption of data to preferably use EnvelopedData.

Replace the text of the section with the following text:

5.2.2. Encrypted Values

Where encrypted data (in this specification, private keys, certificates, or revocation passphrase) is sent in PKI messages, the EncryptedKey data structure is used.

```
EncryptedKey ::= CHOICE {
   encryptedValue        EncryptedValue, -- deprecated
   envelopedData     [0] EnvelopedData }
```

See Certificate Request Message Format (CRMF) [RFC4211] for EncryptedKey and EncryptedValue syntax and Cryptographic Message Syntax (CMS) [RFC5652] for EnvelopedData syntax. Using the EncryptedKey data structure offers the choice to either use EncryptedValue (for backward compatibility only) or EnvelopedData. The use of the EncryptedValue structure has been deprecated in favor of the EnvelopedData structure. Therefore, it is **RECOMMENDED** to use EnvelopedData.

Note: The EncryptedKey structure defined in CRMF [RFC4211] is reused here, which makes the update backward compatible. Using the new syntax with the untagged default choice EncryptedValue is bits-on-the-wire compatible with the old syntax.

To indicate support for EnvelopedData, the pvno cmp2021 has been introduced. Details on the usage of the protocol version number (pvno) are described in Section 7.

The EncryptedKey data structure is used in CMP to transport a private key, certificate, or revocation passphrase in encrypted form.

EnvelopedData is used as follows:

- It contains only one RecipientInfo structure because the content is encrypted only for one recipient.
- It may contain a private key in the AsymmetricKeyPackage structure, as defined in [RFC5958], that is wrapped in a SignedData structure, as specified in Section 5 of CMS [RFC5652] and [RFC8933], and signed by the Key Generation Authority.
- It may contain a certificate or revocation passphrase directly in the encryptedContent field.

The content of the EnvelopedData structure, as specified in Section 6 of CMS [RFC5652], **MUST** be encrypted using a newly generated symmetric content-encryption key. This content-encryption key **MUST** be securely provided to the recipient using one of three key management techniques.

The choice of the key management technique to be used by the sender depends on the credential available at the recipient:

- recipient's certificate with an algorithm identifier and a public key that supports key transport and where any given key usage extension allows keyEncipherment: The content-encryption key will be protected using the key transport key management technique, as specified in Section 6.2.1 of CMS [RFC5652].
- recipient's certificate with an algorithm identifier and a public key that supports key agreement and where any given key usage extension allows keyAgreement: The content-encryption key will be protected using the key agreement key management technique, as specified in Section 6.2.2 of CMS [RFC5652].
- a password or shared secret: The content-encryption key will be protected using the password-based key management technique, as specified in Section 6.2.4 of CMS [RFC5652].

2.8. New Section 5.2.9 - GeneralizedTime

The following subsection points implementers to [RFC5280] regarding usage of GeneralizedTime.

Insert this section after Section 5.2.8.4 of [RFC4210]:

5.2.9 GeneralizedTime

GeneralizedTime is a standard ASN.1 type and **SHALL** be used as specified in Section 4.1.2.5.2 of [RFC5280].

2.9. Update Section 5.3.4 - Certification Response

Section 5.3.4 of [RFC4210] describes the Certification Response. This document updates the syntax by using the parent structure EncryptedKey instead of EncryptedValue, as described in Section 2.7 above. Additionally, it clarifies the certReqId to be used in response to a p10cr message.

Replace the ASN.1 syntax with the following text (Note: This also fixes Errata ID 3949 and 4078):

```
CertRepMessage ::= SEQUENCE {
    caPubs          [1] SEQUENCE SIZE (1..MAX) OF CMPCertificate
                        OPTIONAL,
    response            SEQUENCE OF CertResponse
}

CertResponse ::= SEQUENCE {
    certReqId           INTEGER,
    status              PKIStatusInfo,
    certifiedKeyPair    CertifiedKeyPair    OPTIONAL,
    rspInfo             OCTET STRING        OPTIONAL
    -- analogous to the id-regInfo-utf8Pairs string defined
    -- for regInfo in CertReqMsg [RFC4211]
}

CertifiedKeyPair ::= SEQUENCE {
   certOrEncCert       CertOrEncCert,
   privateKey      [0] EncryptedKey        OPTIONAL,
   -- See [RFC4211] for comments on encoding.
   publicationInfo [1] PKIPublicationInfo  OPTIONAL
}

CertOrEncCert ::= CHOICE {
   certificate     [0] CMPCertificate,
   encryptedCert   [1] EncryptedKey
}
```

Add the following as a new paragraph right after the ASN.1 syntax:

A p10cr message contains exactly one CertificationRequestInfo data structure, as specified in PKCS #10 [RFC2986], but no certReqId. Therefore, the certReqId in the corresponding Certification Response (cp) message **MUST** be set to -1.

Add the following as new paragraphs to the end of the section:

The use of EncryptedKey is described in Section 5.2.2.

Note: To indicate support for EnvelopedData, the pvno cmp2021 has been introduced. Details on the usage of different protocol version numbers (pvno) are described in Section 7.

2.10. Update Section 5.3.18 - Certificate Confirmation Content

This section introduces an optional hashAlg field to the CertStatus type used in certConf messages to explicitly specify the hash algorithm for those certificates where no hash algorithm is specified in the signatureAlgorithm field.

Replace the ASN.1 Syntax of CertStatus with the following text:

```
CertStatus ::= SEQUENCE {
   certHash    OCTET STRING,
   certReqId   INTEGER,
   statusInfo  PKIStatusInfo OPTIONAL,
   hashAlg [0] AlgorithmIdentifier{DIGEST-ALGORITHM, {...}}
               OPTIONAL
}
```

The hashAlg field **SHOULD** be used only in exceptional cases where the signatureAlgorithm of the certificate to be confirmed does not specify a hash algorithm in the OID or in the parameters. In such cases, e.g., for EdDSA, the hashAlg **MUST** be used to specify the hash algorithm to be used for calculating the certHash value. Otherwise, the certHash value **SHALL** be computed using the same hash algorithm as used to create and verify the certificate signature. If hashAlg is used, the CMP version indicated by the certConf message header must be cmp2021(3).

2.11. Update Section 5.3.19.2 - Signing Key Pair Types

The following section clarifies the usage of the Signing Key Pair Types on referencing elliptic curves.

Insert this note at the end of Section 5.3.19.2 of [RFC4210]:

Note: In case several elliptic curves are supported, several id-ecPublicKey elements as defined in [RFC5480] need to be given, one per named curve.

2.12. Update Section 5.3.19.3 - Encryption/Key Agreement Key Pair Types

The following section clarifies the use of the Encryption/Key Agreement Key Pair Types on referencing elliptic curves.

Insert this note at the end of Section 5.3.19.3 of [RFC4210]:

Note: In case several elliptic curves are supported, several id-ecPublicKey elements as defined in [RFC5480] need to be given, one per named curve.

2.13. Replace Section 5.3.19.9 - Revocation Passphrase

Section 5.3.19.9 of [RFC4210] describes the provisioning of a revocation passphrase for authenticating a later revocation request. This document updates the handling by using the parent structure EncryptedKey instead of EncryptedValue to transport this information, as described in Section 2.7 above.

Replace the text of the section with the following text:

5.3.19.9. Revocation Passphrase

This **MAY** be used by the EE to send a passphrase to a CA/RA for the purpose of authenticating a later revocation request (in the case that the appropriate signing private key is no longer available to authenticate the request). See Appendix B for further details on the use of this mechanism.

```
GenMsg:    {id-it 12}, EncryptedKey
GenRep:    {id-it 12}, < absent >
```

The use of EncryptedKey is described in Section 5.2.2.

2.14. New Section 5.3.19.14 - CA Certificates

The following subsection describes PKI general messages using id-it-caCerts. The intended use is specified in Section 4.3 of the Lightweight CMP Profile [RFC9483].

Insert this section after Section 5.3.19.13 of [RFC4210]:

5.3.19.14. CA Certificates

This **MAY** be used by the client to get CA certificates.

```
GenMsg:    {id-it 17}, < absent >
GenRep:    {id-it 17}, SEQUENCE SIZE (1..MAX) OF
                       CMPCertificate | < absent >
```

2.15. New Section 5.3.19.15 - Root CA Certificate Update

The following subsection describes PKI general messages using id-it-rootCaCert and id-it-rootCaKeyUpdate. The use is specified in Section 4.3 of the Lightweight CMP Profile [RFC9483].

Insert this section after the new Section 5.3.19.14:

5.3.19.15. Root CA Certificate Update

This **MAY** be used by the client to get an update of a root CA certificate, which is provided in the body of the request message. In contrast to the ckuann message, this approach follows the request/response model.

The EE **SHOULD** reference its current trust anchor in a TrustAnchor structure in the request body, giving the root CA certificate if available; otherwise, the public key value of the trust anchor is given.

```
GenMsg:    {id-it 20}, RootCaCertValue | < absent >
GenRep:    {id-it 18}, RootCaKeyUpdateContent | < absent >

RootCaCertValue ::= CMPCertificate

RootCaKeyUpdateValue ::= RootCaKeyUpdateContent

RootCaKeyUpdateContent ::= SEQUENCE {
   newWithNew       CMPCertificate,
   newWithOld   [0] CMPCertificate OPTIONAL,
   oldWithNew   [1] CMPCertificate OPTIONAL
   }
```

Note: In contrast to CAKeyUpdAnnContent, this type offers omitting newWithOld and oldWithNew in the GenRep message, depending on the needs of the EE.

2.16. New Section 5.3.19.16 - Certificate Request Template

The following subsection introduces the PKI general message using id-it-certReqTemplate. Details are specified in Section 4.3 of the Lightweight CMP Profile [RFC9483].

Insert this section after the new Section 5.3.19.15:

5.3.19.16. Certificate Request Template

This **MAY** be used by the client to get a template containing requirements for certificate request attributes and extensions. The controls id-regCtrl-algId and id-regCtrl-rsaKeyLen **MAY** contain details on the types of subject public keys the CA is willing to certify.

The id-regCtrl-algId control **MAY** be used to identify a cryptographic algorithm (see Section 4.1.2.7 of [RFC5280]) other than rsaEncryption. The algorithm field **SHALL** identify a cryptographic algorithm. The contents of the optional parameters field will vary according to the algorithm identified. For example, when the algorithm is set to id-ecPublicKey, the parameters identify the elliptic curve to be used; see [RFC5480].

The id-regCtrl-rsaKeyLen control **SHALL** be used for algorithm rsaEncryption and **SHALL** contain the intended modulus bit length of the RSA key.

```
GenMsg:    {id-it 19}, < absent >
GenRep:    {id-it 19}, CertReqTemplateContent | < absent >

CertReqTemplateValue  ::= CertReqTemplateContent

CertReqTemplateContent ::= SEQUENCE {
  certTemplate          CertTemplate,
  keySpec               Controls OPTIONAL }

Controls  ::= SEQUENCE SIZE (1..MAX) OF AttributeTypeAndValue

id-regCtrl-algId OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) dod(6) internet(1) security(5)
   mechanisms(5) pkix(7) pkip(5) regCtrl(1) 11 }

AlgIdCtrl ::= AlgorithmIdentifier{ALGORITHM, {...}}

id-regCtrl-rsaKeyLen OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) dod(6) internet(1) security(5)
   mechanisms(5) pkix(7) pkip(5) regCtrl(1) 12 }

RsaKeyLenCtrl ::= INTEGER (1..MAX)
```

The CertReqTemplateValue contains the prefilled certTemplate to be used for a future certificate request. The publicKey field in the certTemplate **MUST NOT** be used. In case the PKI management entity wishes to specify supported public-key algorithms, the keySpec field **MUST** be used. One AttributeTypeAndValue per supported algorithm or RSA key length **MUST** be used.

Note: The controls ASN.1 type is defined in Section 6 of CRMF [RFC4211].

2.17. New Section 5.3.19.17 - CRL Update Retrieval

The following subsection introduces the PKI general message using id-it-crlStatusList and id-it-crls. Details are specified in Section 4.3 of the Lightweight CMP Profile [RFC9483]. Insert this section after the new Section 5.3.19.16:

5.3.19.17. CRL Update Retrieval

This **MAY** be used by the client to get new CRLs, specifying the source of the CRLs and the thisUpdate value of the latest CRL it already has, if available. A CRL source is given either by a DistributionPointName or the GeneralNames of the issuing CA. The DistributionPointName should be treated as an internal pointer to identify a CRL that the server already has and not as a way to ask the server to fetch CRLs from external locations. The server shall only provide those CRLs that are more recent than the ones indicated by the client.

```
GenMsg:    {id-it 22}, SEQUENCE SIZE (1..MAX) OF CRLStatus
GenRep:    {id-it 23}, SEQUENCE SIZE (1..MAX) OF
                          CertificateList  |  < absent >

CRLSource ::= CHOICE {
   dpn          [0] DistributionPointName,
   issuer       [1] GeneralNames }

CRLStatus ::= SEQUENCE {
   source       CRLSource,
   thisUpdate   Time OPTIONAL }
```

2.18. Update Section 5.3.21 - Error Message Content

Section 5.3.21 of [RFC4210] describes the regular use of error messages. This document adds a use by a PKI management entity to initiate delayed delivery in response to certConf, rr, and genm requests and to error messages.

Replace the first sentence of the first paragraph with the following one:

This data structure **MAY** be used by an EE, CA, or RA to convey error information and by a PKI management entity to initiate delayed delivery of responses.

Replace the second paragraph with the following text:

This message **MAY** be generated at any time during a PKI transaction. If the client sends this request, the server **MUST** respond with a PKIConfirm response or another ErrorMsg if any part of the header is not valid. In case a PKI management entity sends an error message to the EE with the pKIStatusInfo field containing the status "waiting", the EE will initiate polling as described in Section 5.3.22. Otherwise, both sides **MUST** treat this message as the end of the transaction (if a transaction is in progress).

2.19. Replace Section 5.3.22 - Polling Request and Response

Section 5.3.22 of [RFC4210] describes when and how polling messages are used for ir, cr, and kur messages. This document extends the polling mechanism for outstanding responses to any kind of request message. This update also fixes the inconsistent use of the terms 'pReq' vs. 'pollReq' and 'pRep' vs. 'pollRep'.

Replace Section 5.3.22 of [RFC4210] with following text:

This pair of messages is intended to handle scenarios in which the client needs to poll the server to determine the status of an outstanding response (i.e., when the "waiting" PKIStatus has been received).

```
PollReqContent ::= SEQUENCE OF SEQUENCE {
   certReqId    INTEGER }

PollRepContent ::= SEQUENCE OF SEQUENCE {
   certReqId    INTEGER,
   checkAfter   INTEGER,  -- time in seconds
   reason       PKIFreeText OPTIONAL }
```

In response to an ir, cr, p10cr, or kur request message, polling is initiated with an ip, cp, or kup response message containing status "waiting". For any type of request message, polling can be initiated with an error response messages with status "waiting". The following clauses describe how polling messages are used. It is assumed that multiple certConf messages can be sent during transactions. There will be one sent in response to each ip, cp, or kup that contains a CertStatus for an issued certificate.

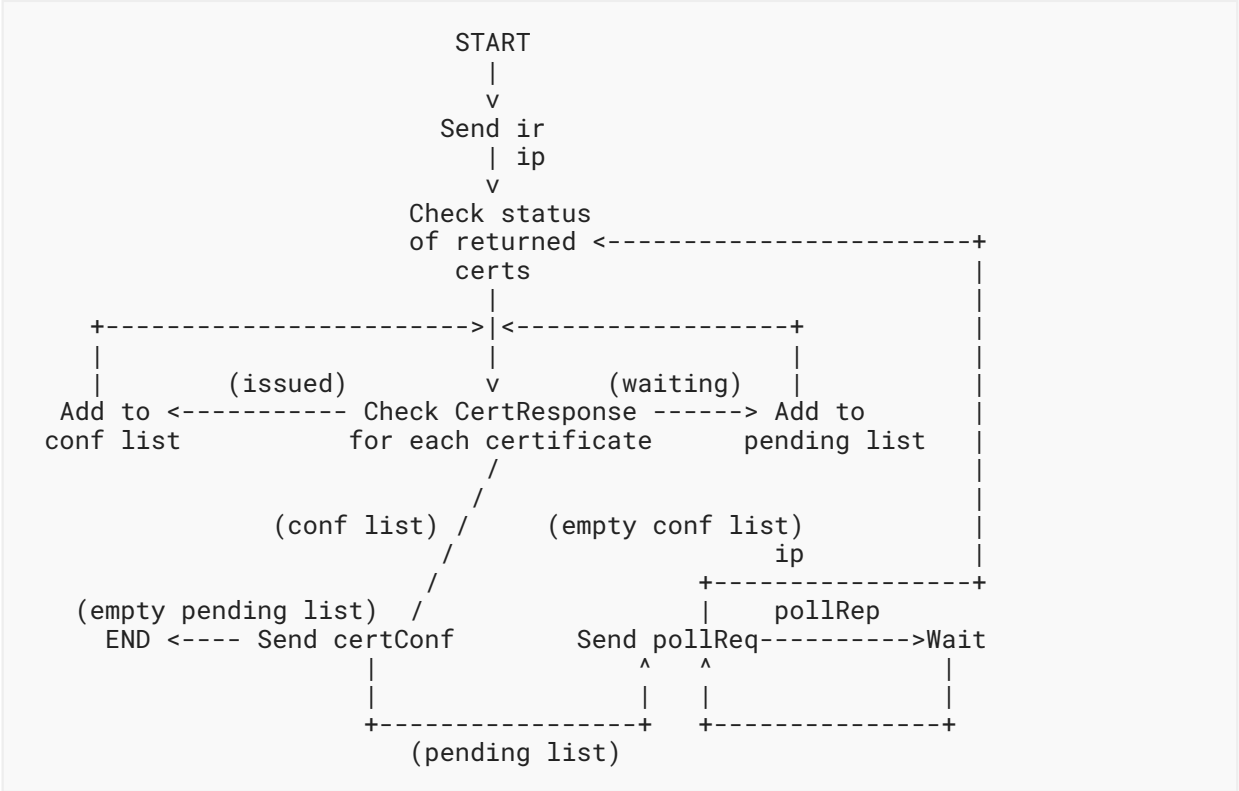
1 In response to an ip, cp, or kup message, an EE will send a certConf for all issued certificates and expect a PKIconf for each certConf. An EE will send a pollReq message in response to each CertResponse element of an ip, cp, or kup message with status "waiting" and in response to an error message with status "waiting". Its certReqId **MUST** be either the index of a CertResponse data structure with status "waiting" or -1, referring to the complete response.

2 In response to a pollReq, a CA/RA will return an ip, cp, or kup if one or more of the still pending requested certificates are ready or the final response to some other type of request is available; otherwise, it will return a pollRep.

3 If the EE receives a pollRep, it will wait for at least the number of seconds given in the checkAfter field before sending another pollReq.

4 If the EE receives an ip, cp, or kup, then it will be treated in the same way as the initial response; if it receives any other response, then this will be treated as the final response to the original request.

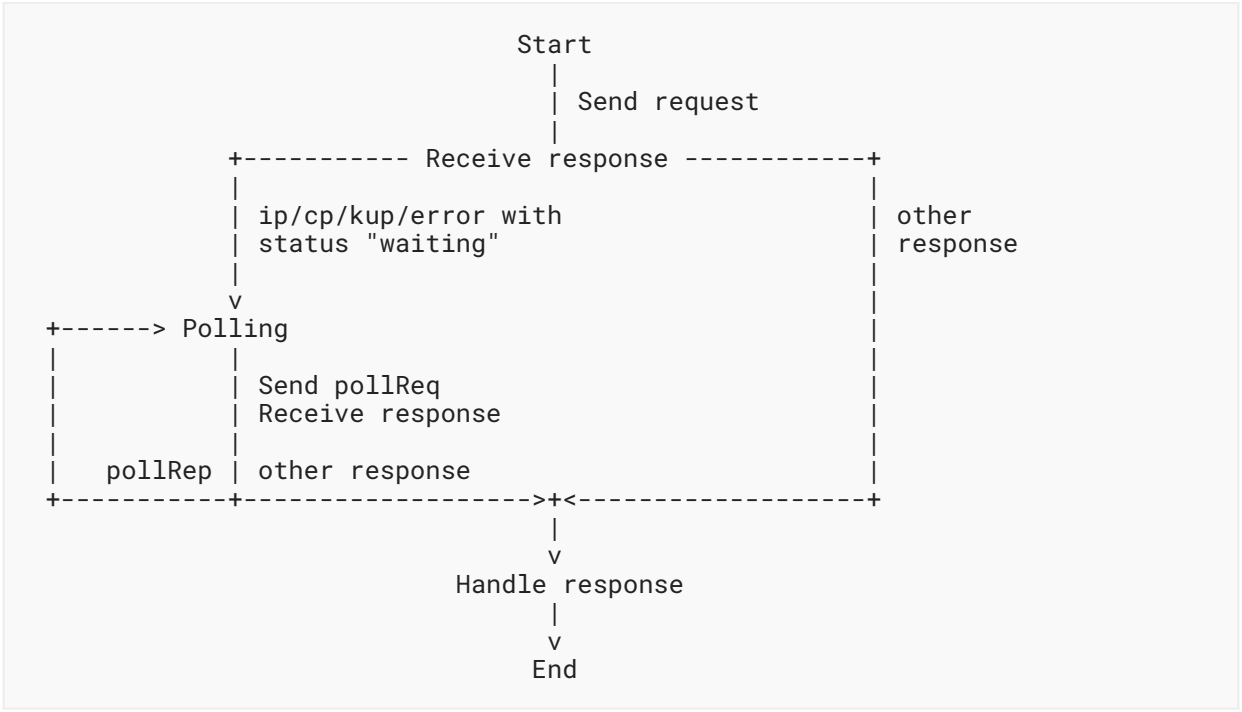
The following client-side state machine describes polling for individual CertResponse elements.



In the following exchange, the end entity is enrolling for two certificates in one request.

```
Step  End Entity                       PKI
-------------------------------------------------------------------
1   Format ir
2                    -> ir      ->
3                                    Handle ir
4                                    Manual intervention is
                                     required for both certs
5                    <- ip      <-
6   Process ip
7   Format pollReq
8                    -> pollReq  ->
9                                    Check status of cert requests
10                                   Certificates not ready
11                                   Format pollRep
12                   <- pollRep  <-
13  Wait
14  Format pollReq
15                   -> pollReq  ->
16                                   Check status of cert requests
17                                   One certificate is ready
18                                   Format ip
19                   <- ip       <-
20  Handle ip
21  Format certConf
22                   -> certConf ->
23                                   Handle certConf
24                                   Format ack
25                   <- pkiConf   <-
26  Format pollReq
27                   -> pollReq  ->
28                                   Check status of certificate
29                                   Certificate is ready
30                                   Format ip
31                   <- ip       <-
31  Handle ip
32  Format certConf
33                   -> certConf ->
34                                   Handle certConf
35                                   Format ack
36                   <- pkiConf  <-
```

The following client-side state machine describes polling for a complete response message.



In the following exchange, the end entity is sending a general message request, and the response is delayed by the server.

```
Step  End Entity                        PKI
-------------------------------------------------------------------
1   Format genm
2                  -> genm     ->
3                                 Handle genm
4                                 delay in response is necessary
5                                 Format error message "waiting"
                                  with certReqId set to -1
6                   <- error   <-
7   Process error
8   Format pollReq
9                   -> pollReq ->
10                                Check status of original request
                                  general message response not ready
11                                Format pollRep
12                  <- pollRep <-
13  Wait
14  Format pollReq
15                  -> pollReq ->
16                                Check status of original request
                                  general message response is ready
17                                Format genp
18                  <- genp    <-
19  Handle genp
```

2.20. Update Section 7 - Version Negotiation

Section 7 of [RFC4210] describes the use of CMP versions. This document describes the handling of the additional CMP version cmp2021, which is introduced to indicate support of EnvelopedData and hashAlg.

Replace the text of the second paragraph with the following text:

If a client knows the protocol version(s) supported by the server (e.g., from a previous PKIMessage exchange or via some out-of-band means), then it **MUST** send a PKIMessage with the highest version supported by both it and the server. If a client does not know what version(s) the server supports, then it **MUST** send a PKIMessage using the highest version it supports with the following exception. Version cmp2021 **SHOULD** only be used if cmp2021 syntax is needed for the request being sent or for the expected response.

Note: Using cmp2000 as the default pvno is done to avoid extra message exchanges for version negotiation and to foster compatibility with cmp2000 implementations. Version cmp2021 syntax is only needed if a message exchange uses hashAlg (in CertStatus) or EnvelopedData.

2.21. Update Section 7.1.1 - Clients Talking to RFC 2510 Servers

Section 7.1.1 of [RFC4210] describes the behavior of a client sending a cmp2000 message talking to a cmp1999 server, as specified in [RFC2510]. This document extends the section to clients with any higher version than cmp1999.

Replace the first sentence of Section 7.1.1 of [RFC4210] with the following text:

If, after sending a message with a protocol version number higher than cmp1999, a client receives an ErrorMsgContent with a version of cmp1999, then it **MUST** abort the current transaction.

2.22. Add Section 8.4 - Private Keys for Certificate Signing and CMP Message Protection

The following subsection addresses the risk arising from reusing the CA private key for CMP message protection.

Insert this section after Section 8.3 of [RFC4210] (Note: This fixes Errata ID 5731):

8.4. Private Keys for Certificate Signing and CMP Message Protection

A CA should not reuse its certificate signing key for other purposes, such as protecting CMP responses and TLS connections. This way, exposure to other parts of the system and the number of uses of this particularly critical key are reduced to a minimum.

2.23. Add Section 8.5 - Entropy of Random Numbers, Key Pairs, and Shared Secret Information

The following subsection addresses the risk arising from low entropy of random numbers, asymmetric keys, and shared secret information.

Insert this section after the new Section 8.4:

8.5. Entropy of Random Numbers, Key Pairs, and Shared Secret Information

Implementations must generate nonces and private keys from random input. The use of inadequate pseudorandom number generators (PRNGs) to generate cryptographic keys can result in little or no security. An attacker may find it much easier to reproduce the PRNG environment that produced the keys and to search the resulting small set of possibilities than brute-force searching the whole key space. As an example of predictable random numbers, see [CVE-2008-0166]; consequences of low-entropy random numbers are discussed in Mining Your Ps and Qs [MiningPsQs]. The generation of quality random numbers is difficult. ISO/IEC 20543:2019 [ISO.20543-2019], NIST SP 800-90A Rev.1 [NIST_SP_800_90Ar1], BSI AIS 31 V2.0 [AIS31], and other specifications offer valuable guidance in this area.

If shared secret information is generated by a cryptographically secure random number generator (CSRNG), it is safe to assume that the entropy of the shared secret information equals its bit length. If no CSRNG is used, the entropy of shared secret information depends on the details of the generation process and cannot be measured securely after it has been generated. If user-generated passwords are used as shared secret information, their entropy cannot be measured and are typically insufficient for protected delivery of centrally generated keys or trust anchors.

If the entropy of shared secret information protecting the delivery of a centrally generated key pair is known, it should not be less than the security strength of that key pair; if the shared secret information is reused for different key pairs, the security of the shared secret information should exceed the security strength of each individual key pair.

For the case of a PKI management operation that delivers a new trust anchor (e.g., a root CA certificate) using caPubs or genm that is (a) not concluded in a timely manner or (b) where the shared secret information is reused for several key management operations, the entropy of the shared secret information, if known, should not be less than the security strength of the trust anchor being managed by the operation. The shared secret information should have an entropy that at least matches the security strength of the key material being managed by the operation. Certain use cases may require shared secret information that may be of a low security strength, e.g., a human-generated password. It is **RECOMMENDED** that such secret information be limited to a single PKI management operation.

2.24. Add Section 8.6 - Trust Anchor Provisioning Using CMP Messages

The following subsection addresses the risk arising from in-band provisioning of new trust anchors in a PKI management operation.

Insert this section after the new Section 8.5:

8.6. Trust Anchor Provisioning Using CMP Messages

A provider of trust anchors, which may be an RA involved in configuration management of its clients, **MUST NOT** include to-be-trusted CA certificates in a CMP message unless the specific deployment scenario can ensure that it is adequate that the receiving EE trusts these certificates, e.g., by loading them into its trust store.

Whenever an EE receives in a CMP message a CA certificate to be used as a trust anchor (for example in the caPubs field of a certificate response or in a general response), it **MUST** properly authenticate the message sender with existing trust anchor information without requiring the new trust anchors included in the message.

Additionally, the EE **MUST** verify that the sender is an authorized source of trust anchors. This authorization is governed by local policy and typically indicated using shared secret information or with a signature-based message protection using a certificate issued by a PKI that is explicitly authorized for this purpose.

2.25. Add Section 8.7 - Authorizing Requests for Certificates with Specific EKUs

The following subsection addresses the security considerations to follow when authorizing requests for certificates containing specific EKUs.

Insert this section after new Section 8.6:

8.7. Authorizing Requests for Certificates with Specific EKUs

When a CA issues a certificate containing extended key usage extensions as defined in Section 4.5, this expresses delegation of an authorization that originally is only with the CA certificate itself. Such delegation is a very sensitive action in a PKI and therefore special care must be taken when approving such certificate requests to ensure that only legitimate entities receive a certificate containing such an EKU.

2.26. Update Appendix B - The Use of Revocation Passphrase

Appendix B of [RFC4210] describes the use of the revocation passphrase. As this document updates [RFC4210] to utilize the parent structure EncryptedKey instead of EncryptedValue as described in Section 2.7 above, the description is updated accordingly.

Replace the first bullet point of this section with the following text:

- The OID and value specified in Section 5.3.19.9 **MAY** be sent in a GenMsg message at any time or **MAY** be sent in the generalInfo field of the PKIHeader of any PKIMessage at any time. (In particular, the EncryptedKey structure as described in Section 5.2.2 may be sent in the header of the certConf message that confirms acceptance of certificates requested in an initialization request or certificate request message.) This conveys a revocation passphrase chosen by the entity to the relevant CA/RA. When EnvelopedData is used, this is in the decrypted bytes of the encryptedContent field. When EncryptedValue is used, this is in the decrypted bytes of the encValue field. Furthermore, the transfer is accomplished with appropriate confidentiality characteristics.

Replace the third bullet point of this section with the following text:

- Either the localKeyId attribute of EnvelopedData as specified in [RFC2985] or the valueHint field of EncryptedValue **MAY** contain a key identifier (chosen by the entity, along with the passphrase itself) to assist in later retrieval of the correct passphrase (e.g., when the revocation request is constructed by the entity and received by the CA/RA).

2.27. Update Appendix C - Request Message Behavioral Clarifications

Appendix C of [RFC4210] provides clarifications to the request message behavior. As this document updates [RFC4210] to utilize the parent structure EncryptedKey instead of EncryptedValue as described in Section 2.7 above, the description is updated accordingly.

Replace the comment within the ASN.1 syntax coming after the definition of POPOSigningKey with the following text (Note: This fixes Errata ID 2615):

```
-- **********
-- * For the purposes of this specification, the ASN.1 comment
-- * given in [RFC4211] pertains not only to certTemplate but
-- * also to the altCertTemplate control.
-- **********
-- * The signature (using "algorithmIdentifier") is on the
-- * DER-encoded value of poposkInput (i.e., the "value" OCTETs
-- * of the POPOSigningKeyInput DER).  NOTE: If CertReqMsg
-- * certReq certTemplate (or the altCertTemplate control)
-- * contains the subject and publicKey values, then poposkInput
-- * MUST be omitted and the signature MUST be computed on the
-- * DER-encoded value of CertReqMsg certReq (or the DER-
-- * encoded value of AltCertTemplate).  If
-- * certTemplate/altCertTemplate does not contain both the
-- * subject and public key values (i.e., if it contains only
-- * one of these or neither), then poposkInput MUST be present
-- * and MUST be signed.
-- **********
```

Replace the ASN.1 syntax of POPOPrivKey with the following text:

```
POPOPrivKey ::= CHOICE {
    thisMessage       [0] BIT STRING,   -- deprecated
    subsequentMessage [1] SubsequentMessage,
    dhMAC             [2] BIT STRING,   -- deprecated
    agreeMAC          [3] PKMACValue,
    encryptedKey      [4] EnvelopedData }
-- **********
-- * When using CMP V2, the encrypted value MUST be transferred in
-- * the thisMessage field that is given as BIT STRING in [RFC4211],
-- * but it requires EncryptedValue. Therefore, this document makes
-- * the behavioral clarification for CMP V2 of specifying that the
-- * contents of "thisMessage" MUST be encoded as an
-- * EncryptedValue and then wrapped in a BIT STRING.
-- * When using CMP V3, the encrypted value MUST be transferred
-- * in the encryptedKey field, as specified in Section 5.2.2.
-- **********
```

2.28. Update Appendix D.1. - General Rules for Interpretation of These Profiles

Appendix D.1 of [RFC4210] provides general rules for interpretation of the PKI management messages profiles specified in Appendices D and E of [RFC4210]. This document updates a sentence regarding the new protocol version cmp2021.

Replace the last sentence of the first paragraph of the section with the following text:

Mandatory fields are not mentioned if they have an obvious value (e.g., in this version of these profiles, pvno is always cmp2000).

2.29. Update Appendix D.2. - Algorithm Use Profile

Appendix D.2 of [RFC4210] provides a list of algorithms that implementations must support when claiming conformance with PKI management message profiles, as specified in Appendix D.2 of CMP [RFC4210]. This document redirects to the new algorithm profile, as specified in Section 7.1 of CMP Algorithms [RFC9481].

Replace the text of the section with the following text:

D.2. Algorithm Use Profile

For specifications of algorithm identifiers and respective conventions for conforming implementations, please refer to Section 7.1 of CMP Algorithms [RFC9481].

2.30. Update Appendix D.4. - Initial Registration/Certification (Basic Authenticated Scheme)

Appendix D.4 of [RFC4210] provides the initial registration/certification scheme. This scheme shall continue using EncryptedValue for backward compatibility reasons.

Replace the line specifying protectionAlg of the Initialization Response message with the following text (Note: This fixes Errata ID 5201):

```
protectionAlg        MSG_MAC_ALG
```

Replace the comment after the privateKey field of crc[1].certifiedKeyPair in the syntax of the Initialization Response message with the following text:

```
   -- see Appendix C (Request Message Behavioral Clarifications)
   -- for backward compatibility reasons, use EncryptedValue
```

3. Updates to RFC 6712 - HTTP Transfer for the Certificate Management Protocol (CMP)

3.1. Update Section 1 - Introduction

To indicate and explain why delayed delivery of all kinds of PKIMessages may be handled at transfer level and/or at CMP level, the introduction of [RFC6712] is updated.

Replace the third paragraph of this section with the following text:

In addition to reliable transport, CMP requires connection and error handling from the transfer protocol, which is all covered by HTTP. Additionally, delayed delivery of CMP response messages may be handled at transfer level, regardless of the message contents. Since this document extends the polling mechanism specified in the second version of CMP [RFC4210] to cover all types of PKI management transactions, delays detected at application level may also be handled within CMP, using pollReq and pollRep messages.

3.2. New Section 1.1 - Changes Since RFC 6712

The following subsection describes feature updates to [RFC6712]. They are related to the base specification. Hence, references to the original sections in [RFC6712] are used whenever possible.

Insert this section after the current Section 1 of [RFC6712]:

1.1 Changes Since RFC 6712

The following updates are made in this document:

- Introduce the HTTP path '/.well-known/cmp'.
- Extend the URI structure.

3.3. Replace Section 3.6 - HTTP Request-URI

Section 3.6 of [RFC6712] specifies the used HTTP URIs. This document introduces the HTTP path '/.well-known/cmp' and extends the URIs.

Replace the text of the section with the following text:

3.6. HTTP Request-URI

Each CMP server on a PKI management entity supporting HTTP or HTTPS transfer **MUST** support the use of the path prefix '/.well-known/' as defined in [RFC8615] and the registered name 'cmp' to ease interworking in a multi-vendor environment.

The CMP client needs to be configured with sufficient information to form the CMP server URI. This is at least the authority portion of the URI, e.g., 'www.example.com:80', or the full operation path segment of the PKI management entity. Additionally, **OPTIONAL** path segments **MAY** be added after the registered application name as part of the full operation path to provide further distinction. The path segment 'p' followed by an arbitraryLabel <name> could, for example, support the differentiation of specific CAs or certificate profiles. Further path segments, e.g., as specified in the Lightweight CMP Profile [RFC9483], could indicate PKI management operations using an operationLabel <operation>. A valid, full CMP URI can look like this:

http://www.example.com/.well-known/cmp

http://www.example.com/.well-known/cmp/<operation>

http://www.example.com/.well-known/cmp/p/<name>

http://www.example.com/.well-known/cmp/p/<name>/<operation>

4. IANA Considerations

4.1. Updates to the ASN.1 Modules in RFCs 4210 and 5912

This document updates the ASN.1 modules of Appendix F of [RFC4210] and Section 9 of [RFC5912] as shown in Appendixes A.1 and A.2 of this document, respectively. The OIDs 99 (id-mod-cmp2021-88) and 100 (id-mod-cmp2021-02) have been registered in the "SMI Security for PKIX Module Identifier" registry to identify the updated ASN.1 modules.

4.2. Updates to the IANA Considerations of RFC 4210

This document updates the IANA Consideration sections of [RFC4210] by adding this content.

4.2.1. SMI Security for PKIX Extended Key Purpose Registry

IANA has registered the following new entry in the "SMI Security for PKIX Extended Key Purpose" registry (see <https://www.iana.org/assignments/smi-numbers>, as defined in [RFC7299]:

Decimal	Description	References
32	id-kp-cmKGA	RFC 9480

Table 1: Addition to the SMI Security for PKIX Extended Key Purpose

4.2.2. SMI Security for PKIX CMP Information Types

IANA has registered the following new entries in the "SMI Security for PKIX CMP Information Types" registry (see <https://www.iana.org/assignments/smi-numbers>), as defined in [RFC7299]:

Decimal	Description	References
17	id-it-caCerts	RFC 9480
18	id-it-rootCaKeyUpdate	RFC 9480
19	id-it-certReqTemplate	RFC 9480
20	id-it-rootCaCert	RFC 9480
21	id-it-certProfile	RFC 9480
22	id-it-crlStatusList	RFC 9480
23	id-it-crls	RFC 9480

Table 2: Additions to the PKIX CMP Information Types Registry

4.2.3. SMI Security for PKIX CRMF Registration Controls

IANA has registered the following new entries in the "SMI Security for PKIX CRMF Registration Controls" registry (see <https://www.iana.org/assignments/smi-numbers>), as defined in [RFC7299]:

Decimal	Description	References
11	id-regCtrl-algId	RFC 9480
12	id-regCtrl-rsaKeyLen	RFC 9480

Table 3: Addition to the PKIX CRMF Registration Controls Registry

4.3. Updates to the IANA Considerations of RFC 6712

This document contains an update to the IANA Considerations sections of [RFC6712] by adding this content.

4.3.1. Well-Known URIs

IANA has registered the following new entry in the "Well-Known URIs" registry (see <https://www.iana.org/assignments/well-known-uris>), as defined in [RFC8615]:

URI Suffix: cmp
Change Controller: IETF
Reference: [RFC9480] [RFC9482]
Status: permanent
Related Information: CMP has a registry at <https://www.iana.org/assignments/cmp>

4.3.2. Certificate Management Protocol (CMP) Registry

This document defines a new protocol registry group entitled "Certificate Management Protocol (CMP)" (at <https://www.iana.org/assignments/cmp>) with a new "CMP Well-Known URI Path Segments" registry containing three columns: Path Segment, Description, and Reference. New items can be added using the Specification Required [RFC8615] process. The initial entry of this registry is:

Path Segment: p
Description: Indicates that the next path segment specifies, e.g., a CA or certificate profile name
Reference: [RFC9480] [RFC9482]

5. Security Considerations

The security considerations of [RFC4210] are extended in Section 2.22 to Section 2.24. No security considerations updates of [RFC6712] were required.

6. References

6.1. 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>.

[RFC2510] Adams, C. and S. Farrell, "Internet X.509 Public Key Infrastructure Certificate Management Protocols", RFC 2510, DOI 10.17487/RFC2510, March 1999, <https://www.rfc-editor.org/info/rfc2510>.

[RFC2985] Nystrom, M. and B. Kaliski, "PKCS #9: Selected Object Classes and Attribute Types Version 2.0", RFC 2985, DOI 10.17487/RFC2985, November 2000, <https://www.rfc-editor.org/info/rfc2985>.

[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>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[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>.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <https://www.rfc-editor.org/info/rfc5480>.

[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>.

[RFC6402] Schaad, J., "Certificate Management over CMS (CMC) Updates", RFC 6402, DOI 10.17487/RFC6402, November 2011, <https://www.rfc-editor.org/info/rfc6402>.

[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>.

[RFC9481] Brockhaus, H., Aschauer, H., Ounsworth, M., and J. Gray, "Certificate Management Protocol (CMP) Algorithms", RFC 9481, DOI 10.17487/RFC9481, October 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, October 2023, <https://www.rfc-editor.org/info/rfc9482>.

6.2. Informative References

[AIS31] Killmann, W. and W. Schindler, "A proposal for: Functionality classes for random number generators - Version 2.0", September 2011, <https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_for_random_number_generators_e.pdf>.

[CVE-2008-0166] National Institute of Science and Technology (NIST), "National Vulnerability Database - CVE-2008-0166", May 2008, <https://nvd.nist.gov/vuln/detail/CVE-2008-0166>.

[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>.

[ISO.20543-2019] International Organization for Standardization (ISO), "Information technology -- Security techniques -- Test and analysis methods for random bit generators within ISO/IEC 19790 and ISO/IEC 15408", ISO/IEC 20543:2019, October 2019.

[MiningPsQs] Heninger, N., Durumeric, Z., Wustrow, E., and J. A. Halderman, "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices", Security'12: Proceedings of the 21st USENIX conference on Security symposium, August 2012, <https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/heninger>.

[NIST_SP_800_90Ar1] Barker, E. B., Kelsey, J. M., and NIST, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", NIST Special Publications (General) 800-90Ar1, DOI 10.6028/NIST.SP.800-90Ar1, June 2015, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf>.

[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>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC2202] Cheng, P. and R. Glenn, "Test Cases for HMAC-MD5 and HMAC-SHA-1", RFC 2202, DOI 10.17487/RFC2202, September 1997, <https://www.rfc-editor.org/info/rfc2202>.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <https://www.rfc-editor.org/info/rfc5912>.

[RFC7299] Housley, R., "Object Identifier Registry for the PKIX Working Group", RFC 7299, DOI 10.17487/RFC7299, July 2014, <https://www.rfc-editor.org/info/rfc7299>.

[RFC9483] Brockhaus, H., Fries, S., and D. von Oheimb, "Lightweight Certificate Management Protocol (CMP) Profile", RFC 9483, DOI 10.17487/RFC9483, October 2023, <https://www.rfc-editor.org/info/rfc9483>.

Appendix A. ASN.1 Modules

A.1. Update to RFC 4210 - 1988 ASN.1 Module

This section contains the updated ASN.1 module for [RFC4210]. This module replaces the module in Appendix F of that document. Although a 2002 ASN.1 module is provided, this 1988 ASN.1 module remains the normative module, as per the policy of the PKIX Working Group.

```
PKIXCMP {iso(1) identified-organization(3)
      dod(6) internet(1) security(5) mechanisms(5) pkix(7)
      id-mod(0) id-mod-cmp2021-88(99)}

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL --

IMPORTS

    Certificate, CertificateList, Extensions, Name, Time,
    AlgorithmIdentifier, id-kp
    --, UTF8String -- -- if required; otherwise, comment out
           FROM PKIX1Explicit88 {iso(1) identified-organization(3)
           dod(6) internet(1) security(5) mechanisms(5) pkix(7)
           id-mod(0) id-pkix1-explicit-88(18)}
    -- The import of Name is added to define CertificationRequest
    -- instead of importing it from PKCS #10 [RFC2986].

    DistributionPointName, GeneralNames, GeneralName, KeyIdentifier
           FROM PKIX1Implicit88 {iso(1) identified-organization(3)
           dod(6) internet(1) security(5) mechanisms(5) pkix(7)
           id-mod(0) id-pkix1-implicit-88(19)}
```

```
    CertTemplate, PKIPublicationInfo, EncryptedKey, CertId,
    CertReqMessages, Controls, AttributeTypeAndValue, id-regCtrl
           FROM PKIXCRMF-2005 {iso(1) identified-organization(3)
           dod(6) internet(1) security(5) mechanisms(5) pkix(7)
           id-mod(0) id-mod-crmf2005(36)}
    -- The import of EncryptedKey is added due to the updates made
    -- in CMP Updates [RFC9480]. EncryptedValue does not need to
    -- be imported anymore and is therefore removed here.

    -- Also, see the behavioral clarifications to CRMF codified in
    -- Appendix C of this specification.

    EnvelopedData, SignedData, Attribute
           FROM CryptographicMessageSyntax2004 { iso(1)
           member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
           smime(16) modules(0) cms-2004(24) }
    -- The import of EnvelopedData and SignedData is added due to
    -- the updates made in CMP Updates [RFC9480].
    -- The import of Attribute is added to define
    -- CertificationRequest instead of importing it from
    -- PKCS #10 [RFC2986].

    ;

-- The rest of the module contains locally defined OIDs and
-- constructs:

CMPCertificate ::= CHOICE {
    x509v3PKCert        Certificate
}
-- This syntax, while bits-on-the-wire compatible with the
-- standard X.509 definition of "Certificate", allows the
-- possibility of future certificate types (such as X.509
-- attribute certificates, card-verifiable
-- certificates, or other kinds of certificates) within this
-- Certificate Management Protocol, should a need ever arise to
-- support such generality.  Those implementations that do not
-- foresee a need to ever support other certificate types MAY, if
-- they wish, comment out the above structure and "uncomment" the
-- following one prior to compiling this ASN.1 module.  (Note that
-- interoperability with implementations that don't do this will be
-- unaffected by this change.)

-- CMPCertificate ::= Certificate

PKIMessage ::= SEQUENCE {
    header           PKIHeader,
    body             PKIBody,
    protection   [0] PKIProtection OPTIONAL,
    extraCerts   [1] SEQUENCE SIZE (1..MAX) OF CMPCertificate
                     OPTIONAL
}

PKIMessages ::= SEQUENCE SIZE (1..MAX) OF PKIMessage

PKIHeader ::= SEQUENCE {
    pvno                INTEGER     { cmp1999(1), cmp2000(2),
```

```
                                        cmp2021(3) },
    sender              GeneralName,
    -- identifies the sender
    recipient           GeneralName,
    -- identifies the intended recipient
    messageTime     [0] GeneralizedTime         OPTIONAL,
    -- time of production of this message (used when the sender
    -- believes that the transport will be "suitable", i.e.,
    -- that the time will still be meaningful upon receipt)
    protectionAlg   [1] AlgorithmIdentifier     OPTIONAL,
    -- algorithm used for the calculation of protection bits
    senderKID       [2] KeyIdentifier           OPTIONAL,
    recipKID        [3] KeyIdentifier           OPTIONAL,
    -- to identify specific keys used for protection
    transactionID   [4] OCTET STRING            OPTIONAL,
    -- identifies the transaction, i.e., this will be the same in
    -- corresponding request, response, certConf, and PKIConf
    -- messages
    senderNonce     [5] OCTET STRING            OPTIONAL,
    recipNonce      [6] OCTET STRING            OPTIONAL,
    -- nonces used to provide replay protection, senderNonce
    -- is inserted by the creator of this message; recipNonce
    -- is a nonce previously inserted in a related message by
    -- the intended recipient of this message.
    freeText        [7] PKIFreeText             OPTIONAL,
    -- this may be used to indicate context-specific instructions
    -- (this field is intended for human consumption)
    generalInfo     [8] SEQUENCE SIZE (1..MAX) OF
                            InfoTypeAndValue    OPTIONAL
    -- this may be used to convey context-specific information
    -- (this field not primarily intended for human consumption)
}

PKIFreeText ::= SEQUENCE SIZE (1..MAX) OF UTF8String
    -- text encoded as a UTF-8 string [RFC3629]

PKIBody ::= CHOICE {       -- message-specific body elements
    ir       [0]  CertReqMessages,        --Initialization Request
    ip       [1]  CertRepMessage,         --Initialization Response
    cr       [2]  CertReqMessages,        --Certification Request
    cp       [3]  CertRepMessage,         --Certification Response
    p10cr    [4]  CertificationRequest,   --imported from [RFC2986]
    popdecc  [5]  POPODecKeyChallContent, --pop Challenge
    popdecr  [6]  POPODecKeyRespContent,  --pop Response
    kur      [7]  CertReqMessages,        --Key Update Request
    kup      [8]  CertRepMessage,         --Key Update Response
    krr      [9]  CertReqMessages,        --Key Recovery Request
    krp      [10] KeyRecRepContent,       --Key Recovery Response
    rr       [11] RevReqContent,          --Revocation Request
    rp       [12] RevRepContent,          --Revocation Response
    ccr      [13] CertReqMessages,        --Cross-Cert. Request
    ccp      [14] CertRepMessage,         --Cross-Cert. Response
    ckuann   [15] CAKeyUpdAnnContent,     --CA Key Update Ann.
    cann     [16] CertAnnContent,         --Certificate Ann.
    rann     [17] RevAnnContent,          --Revocation Ann.
    crlann   [18] CRLAnnContent,          --CRL Announcement
    pkiconf  [19] PKIConfirmContent,      --Confirmation
    nested   [20] NestedMessageContent,   --Nested Message
```

```
    genm     [21] GenMsgContent,          --General Message
    genp     [22] GenRepContent,          --General Response
    error    [23] ErrorMsgContent,        --Error Message
    certConf [24] CertConfirmContent,     --Certificate Confirm
    pollReq  [25] PollReqContent,         --Polling Request
    pollRep  [26] PollRepContent          --Polling Response
}

PKIProtection ::= BIT STRING

ProtectedPart ::= SEQUENCE {
    header    PKIHeader,
    body      PKIBody
}

id-PasswordBasedMac OBJECT IDENTIFIER ::= {1 2 840 113533 7 66 13}
PBMParameter ::= SEQUENCE {
    salt                OCTET STRING,
    -- Note: Implementations MAY wish to limit acceptable sizes
    -- of this string to values appropriate for their environment
    -- in order to reduce the risk of denial-of-service attacks.
    owf                 AlgorithmIdentifier,
    -- AlgId for a One-Way Function (OWF)
    iterationCount      INTEGER,
    -- number of times the OWF is applied
    -- Note: Implementations MAY wish to limit acceptable sizes
    -- of this integer to values appropriate for their environment
    -- in order to reduce the risk of denial-of-service attacks.
    mac                 AlgorithmIdentifier
    -- the MAC AlgId (e.g., HMAC-SHA256, AES-GMAC [RFC9481],
}   -- or HMAC [RFC2104, RFC2202])



id-DHBasedMac OBJECT IDENTIFIER ::= {1 2 840 113533 7 66 30}
DHBMParameter ::= SEQUENCE {
    owf                 AlgorithmIdentifier,
    -- AlgId for a One-Way Function
    mac                 AlgorithmIdentifier
    -- the MAC AlgId (e.g., HMAC-SHA256, AES-GMAC [RFC9481],
}   -- or HMAC [RFC2104, RFC2202])


NestedMessageContent ::= PKIMessages

PKIStatus ::= INTEGER {
    accepted              (0),
    -- you got exactly what you asked for
    grantedWithMods       (1),
    -- you got something like what you asked for; the
    -- requester is responsible for ascertaining the differences
    rejection             (2),
    -- you don't get it, more information elsewhere in the message
    waiting               (3),
    -- the request body part has not yet been processed; expect to
    -- hear more later (note: proper handling of this status
    -- response MAY use the polling req/rep PKIMessages specified
    -- in Section 5.3.22 of [RFC4210]; alternatively, polling in the
```

```
    -- underlying transport layer MAY have some utility in this
    -- regard)
    revocationWarning      (4),
    -- this message contains a warning that a revocation is
    -- imminent
    revocationNotification (5),
    -- notification that a revocation has occurred
    keyUpdateWarning       (6)
    -- update already done for the oldCertId specified in
    -- CertReqMsg
}

PKIFailureInfo ::= BIT STRING {
-- since we can fail in more than one way!
-- More codes may be added in the future if/when required.
    badAlg              (0),
    -- unrecognized or unsupported algorithm identifier
    badMessageCheck     (1),
    -- integrity check failed (e.g., signature did not verify)
    badRequest          (2),
    -- transaction not permitted or supported
    badTime             (3),
    -- messageTime was not sufficiently close to the system time,
    -- as defined by local policy
    badCertId           (4),
    -- no certificate could be found matching the provided criteria
    badDataFormat       (5),
    -- the data submitted has the wrong format
    wrongAuthority      (6),
    -- the authority indicated in the request is different from the
    -- one creating the response token
    incorrectData       (7),
    -- the requester's data is incorrect (for notary services)
    missingTimeStamp    (8),
    -- when the timestamp is missing but should be there
    -- (by policy)
    badPOP              (9),
    -- the proof-of-possession failed
    certRevoked         (10),
    -- the certificate has already been revoked
    certConfirmed       (11),
    -- the certificate has already been confirmed
    wrongIntegrity      (12),
    -- not valid integrity, based on the password instead of the
    -- signature or vice versa
    badRecipientNonce   (13),
    -- not valid recipient nonce, either missing or wrong value
    timeNotAvailable    (14),
    -- the time source of the Time Stamping Authority (TSA) is
    -- not available
    unacceptedPolicy    (15),
    -- the requested TSA policy is not supported by the TSA
    unacceptedExtension (16),
    -- the requested extension is not supported by the TSA
    addInfoNotAvailable (17),
    -- the additional information requested could not be
    -- understood or is not available
    badSenderNonce      (18),
```

```
    -- not valid sender nonce, either missing or wrong size
    badCertTemplate     (19),
    -- not valid cert. template or missing mandatory information
    signerNotTrusted    (20),
    -- signer of the message unknown or not trusted
    transactionIdInUse  (21),
    -- the transaction identifier is already in use
    unsupportedVersion  (22),
    -- the version of the message is not supported
    notAuthorized       (23),
    -- the sender was not authorized to make the preceding
    -- request or perform the preceding action
    systemUnavail       (24),
    -- the request cannot be handled due to system unavailability
    systemFailure       (25),
    -- the request cannot be handled due to system failure
    duplicateCertReq    (26)
    -- the certificate cannot be issued because a duplicate
    -- certificate already exists
}

PKIStatusInfo ::= SEQUENCE {
    status        PKIStatus,
    statusString  PKIFreeText     OPTIONAL,
    failInfo      PKIFailureInfo  OPTIONAL
}

OOBCert ::= CMPCertificate

OOBCertHash ::= SEQUENCE {
    hashAlg     [0] AlgorithmIdentifier     OPTIONAL,
    certId      [1] CertId                  OPTIONAL,
    hashVal         BIT STRING
    -- hashVal is calculated over the DER encoding of the
    -- self-signed certificate with the identifier certID.
}

POPODecKeyChallContent ::= SEQUENCE OF Challenge
-- one Challenge per encryption key certification request (in the
-- same order as these requests appear in CertReqMessages)

Challenge ::= SEQUENCE {
    owf                 AlgorithmIdentifier  OPTIONAL,
    -- MUST be present in the first Challenge; MAY be omitted in
    -- any subsequent Challenge in POPODecKeyChallContent (if
    -- omitted, then the owf used in the immediately preceding
    -- Challenge is to be used)
    witness             OCTET STRING,
    -- the result of applying the One-Way Function (owf) to a
    -- randomly generated INTEGER, A (Note that a different
    -- INTEGER MUST be used for each Challenge.)
    challenge           OCTET STRING
    -- the encryption (under the public key for which the cert.
    -- request is being made) of Rand
}

-- Rand was added in CMP Updates [RFC9480]
```

```
Rand ::= SEQUENCE {
-- Rand is encrypted under the public key to form the challenge
-- in POPODecKeyChallContent
   int                 INTEGER,
   -- the randomly generated INTEGER A (above)
   sender              GeneralName
   -- the sender's name (as included in PKIHeader)
}

POPODecKeyRespContent ::= SEQUENCE OF INTEGER
-- One INTEGER per encryption key certification request (in the
-- same order as these requests appear in CertReqMessages).  The
-- retrieved INTEGER A (above) is returned to the sender of the
-- corresponding Challenge.

CertRepMessage ::= SEQUENCE {
    caPubs       [1] SEQUENCE SIZE (1..MAX) OF CMPCertificate
                     OPTIONAL,
    response         SEQUENCE OF CertResponse
}

CertificationRequest ::= SEQUENCE {
    certificationRequestInfo  SEQUENCE {
      version                   INTEGER,
      subject                   Name,
      subjectPublicKeyInfo      SEQUENCE {
        algorithm                 AlgorithmIdentifier,
        subjectPublicKey          BIT STRING },
      attributes                [0] IMPLICIT SET OF Attribute },
    signatureAlgorithm        AlgorithmIdentifier,
    signature                 BIT STRING
}

CertResponse ::= SEQUENCE {
    certReqId           INTEGER,
    -- to match this response with the corresponding request (a value
    -- of -1 is to be used if certReqId is not specified in the
    -- corresponding request, which can only be a p10cr)
    status              PKIStatusInfo,
    certifiedKeyPair    CertifiedKeyPair    OPTIONAL,
    rspInfo             OCTET STRING        OPTIONAL
    -- analogous to the id-regInfo-utf8Pairs string defined
    -- for regInfo in CertReqMsg [RFC4211]
}

CertifiedKeyPair ::= SEQUENCE {
    certOrEncCert       CertOrEncCert,
    privateKey      [0] EncryptedKey        OPTIONAL,
    -- See [RFC4211] for comments on encoding.
    -- Changed from Encrypted Value to EncryptedKey as a CHOICE of
    -- EncryptedValue and EnvelopedData due to the changes made in
    -- CMP Updates [RFC9480].
    -- Using the choice EncryptedValue is bit-compatible to the
    -- syntax without this change.
    publicationInfo [1] PKIPublicationInfo  OPTIONAL
}

CertOrEncCert ::= CHOICE {
```

```
    certificate     [0] CMPCertificate,
    encryptedCert   [1] EncryptedKey
    -- Changed from Encrypted Value to EncryptedKey as a CHOICE of
    -- EncryptedValue and EnvelopedData due to the changes made in
    -- CMP Updates [RFC9480].
    -- Using the choice EncryptedValue is bit-compatible to the
    -- syntax without this change.
}

KeyRecRepContent ::= SEQUENCE {
    status                  PKIStatusInfo,
    newSigCert          [0] CMPCertificate OPTIONAL,
    caCerts             [1] SEQUENCE SIZE (1..MAX) OF
                                        CMPCertificate OPTIONAL,
    keyPairHist         [2] SEQUENCE SIZE (1..MAX) OF
                                        CertifiedKeyPair OPTIONAL
}

RevReqContent ::= SEQUENCE OF RevDetails

RevDetails ::= SEQUENCE {
    certDetails         CertTemplate,
    -- allows the requester to specify as much as they can about
    -- the cert. for which revocation is requested
    -- (e.g., for cases in which serialNumber is not available)
    crlEntryDetails     Extensions       OPTIONAL
    -- requested crlEntryExtensions
}

RevRepContent ::= SEQUENCE {
    status       SEQUENCE SIZE (1..MAX) OF PKIStatusInfo,
    -- in the same order as was sent in RevReqContent
    revCerts [0] SEQUENCE SIZE (1..MAX) OF CertId
                                        OPTIONAL,
    -- IDs for which revocation was requested
    -- (same order as status)
    crls     [1] SEQUENCE SIZE (1..MAX) OF CertificateList
                                        OPTIONAL
    -- the resulting CRLs (there may be more than one)
}

CAKeyUpdAnnContent ::= SEQUENCE {
    oldWithNew   CMPCertificate, -- old pub signed with new priv
    newWithOld   CMPCertificate, -- new pub signed with old priv
    newWithNew   CMPCertificate  -- new pub signed with new priv
}

CertAnnContent ::= CMPCertificate

RevAnnContent ::= SEQUENCE {
    status              PKIStatus,
    certId              CertId,
    willBeRevokedAt     GeneralizedTime,
    badSinceDate        GeneralizedTime,
    crlDetails          Extensions  OPTIONAL
    -- extra CRL details (e.g., crl number, reason, location, etc.)
}
```

```
CRLAnnContent ::= SEQUENCE OF CertificateList

CertConfirmContent ::= SEQUENCE OF CertStatus

CertStatus ::= SEQUENCE {
   certHash    OCTET STRING,
   -- the hash of the certificate, using the same hash algorithm
   -- as is used to create and verify the certificate signature
   certReqId   INTEGER,
   -- to match this confirmation with the corresponding req/rep
   statusInfo  PKIStatusInfo OPTIONAL,
   hashAlg [0] AlgorithmIdentifier OPTIONAL
   -- the hash algorithm to use for calculating certHash
   -- SHOULD NOT be used in all cases where the AlgorithmIdentifier
   -- of the certificate signature specifies a hash algorithm
}

PKIConfirmContent ::= NULL

-- CertReqTemplateContent, id-regCtrl-algId, id-regCtrl-algId, and
-- id-regCtrl-rsaKeyLen were added in CMP Updates [RFC9480]

CertReqTemplateContent ::= SEQUENCE {
   certTemplate           CertTemplate,
   -- prefilled certTemplate structure elements
   -- The SubjectPublicKeyInfo field in the certTemplate MUST NOT
   -- be used.
   keySpec                Controls OPTIONAL
   -- MAY be used to specify supported algorithms
   -- Controls  ::= SEQUENCE SIZE (1..MAX) OF AttributeTypeAndValue
   -- as specified in CRMF [RFC4211]
   }

id-regCtrl-altCertTemplate OBJECT IDENTIFIER ::= { id-regCtrl 7 }
   AltCertTemplate ::= AttributeTypeAndValue
   -- specifies a template for a certificate other than an X.509v3
   -- public key certificate

id-regCtrl-algId OBJECT IDENTIFIER ::= { id-regCtrl 11 }
   AlgIdCtrl ::= AlgorithmIdentifier
   -- SHALL be used to specify supported algorithms other than RSA

id-regCtrl-rsaKeyLen OBJECT IDENTIFIER ::= { id-regCtrl 12 }
   RsaKeyLenCtrl ::= INTEGER (1..MAX)
   -- SHALL be used to specify supported RSA key lengths

-- RootCaKeyUpdateContent, CRLSource, and CRLStatus were added in
-- CMP Updates [RFC9480]

RootCaKeyUpdateContent ::= SEQUENCE {
   newWithNew       CMPCertificate,
   -- new root CA certificate
   newWithOld   [0] CMPCertificate OPTIONAL,
   -- X.509 certificate containing the new public root CA key
   -- signed with the old private root CA key
   oldWithNew   [1] CMPCertificate OPTIONAL
   -- X.509 certificate containing the old public root CA key
   -- signed with the new private root CA key
```

```
   }

CRLSource ::= CHOICE {
   dpn          [0] DistributionPointName,
   issuer       [1] GeneralNames }

CRLStatus ::= SEQUENCE {
   source       CRLSource,
   thisUpdate   Time OPTIONAL }

InfoTypeAndValue ::= SEQUENCE {
    infoType               OBJECT IDENTIFIER,
    infoValue              ANY DEFINED BY infoType  OPTIONAL
}
-- Example InfoTypeAndValue contents include, but are not limited
-- to, the following (uncomment in this ASN.1 module and use as
-- appropriate for a given environment):
--
--   id-it-caProtEncCert    OBJECT IDENTIFIER ::= {id-it 1}
--      CAProtEncCertValue      ::= CMPCertificate
--   id-it-signKeyPairTypes OBJECT IDENTIFIER ::= {id-it 2}
--      SignKeyPairTypesValue   ::= SEQUENCE SIZE (1..MAX) OF
--                                  AlgorithmIdentifier
--   id-it-encKeyPairTypes  OBJECT IDENTIFIER ::= {id-it 3}
--      EncKeyPairTypesValue    ::= SEQUENCE SIZE (1..MAX) OF
--                                  AlgorithmIdentifier
--   id-it-preferredSymmAlg OBJECT IDENTIFIER ::= {id-it 4}
--      PreferredSymmAlgValue   ::= AlgorithmIdentifier
--   id-it-caKeyUpdateInfo  OBJECT IDENTIFIER ::= {id-it 5}
--      CAKeyUpdateInfoValue    ::= CAKeyUpdAnnContent
--   id-it-currentCRL       OBJECT IDENTIFIER ::= {id-it 6}
--      CurrentCRLValue         ::= CertificateList
--   id-it-unsupportedOIDs  OBJECT IDENTIFIER ::= {id-it 7}
--      UnsupportedOIDsValue    ::= SEQUENCE SIZE (1..MAX) OF
--                                    OBJECT IDENTIFIER
--   id-it-keyPairParamReq  OBJECT IDENTIFIER ::= {id-it 10}
--      KeyPairParamReqValue    ::= OBJECT IDENTIFIER
--   id-it-keyPairParamRep  OBJECT IDENTIFIER ::= {id-it 11}
--      KeyPairParamRepValue    ::= AlgorithmIdentifier
--   id-it-revPassphrase    OBJECT IDENTIFIER ::= {id-it 12}
--      RevPassphraseValue      ::= EncryptedKey
--      - Changed from Encrypted Value to EncryptedKey as a CHOICE
--      - of EncryptedValue and EnvelopedData due to the changes
--      - made in CMP Updates [RFC9480].
--      - Using the choice EncryptedValue is bit-compatible to the
--      - syntax without this change.
--   id-it-implicitConfirm  OBJECT IDENTIFIER ::= {id-it 13}
--      ImplicitConfirmValue    ::= NULL
--   id-it-confirmWaitTime  OBJECT IDENTIFIER ::= {id-it 14}
--      ConfirmWaitTimeValue    ::= GeneralizedTime
--   id-it-origPKIMessage   OBJECT IDENTIFIER ::= {id-it 15}
--      OrigPKIMessageValue     ::= PKIMessages
--   id-it-suppLangTags     OBJECT IDENTIFIER ::= {id-it 16}
--      SuppLangTagsValue       ::= SEQUENCE OF UTF8String
--   id-it-caCerts          OBJECT IDENTIFIER ::= {id-it 17}
--      CaCertsValue            ::= SEQUENCE SIZE (1..MAX) OF
--                                      CMPCertificate
--      - id-it-caCerts added in CMP Updates [RFC9480]
```

```
--   id-it-rootCaKeyUpdate  OBJECT IDENTIFIER ::= {id-it 18}
--      RootCaKeyUpdateValue    ::= RootCaKeyUpdateContent
--      - id-it-rootCaKeyUpdate added in CMP Updates [RFC9480]
--   id-it-certReqTemplate  OBJECT IDENTIFIER ::= {id-it 19}
--      CertReqTemplateValue    ::= CertReqTemplateContent
--      - id-it-certReqTemplate added in CMP Updates [RFC9480]
--   id-it-rootCaCert       OBJECT IDENTIFIER ::= {id-it 20}
--      RootCaCertValue         ::= CMPCertificate
--      - id-it-rootCaCert added in CMP Updates [RFC9480]
--   id-it-certProfile      OBJECT IDENTIFIER ::= {id-it 21}
--      CertProfileValue        ::= SEQUENCE SIZE (1..MAX) OF
--                                                 UTF8String
--      - id-it-certProfile added in CMP Updates [RFC9480]
--   id-it-crlStatusList    OBJECT IDENTIFIER ::= {id-it 22}
--   CRLStatusListValue         ::= SEQUENCE SIZE (1..MAX) OF
--                                                  CRLStatus
--      - id-it-crlStatusList added in CMP Updates [RFC9480]
--   id-it-crls             OBJECT IDENTIFIER ::= {id-it 23}
--   CRLsValue                  ::= SEQUENCE SIZE (1..MAX) OF
--                                                CertificateList
--      - id-it-crls added in CMP Updates [RFC9480]
--
-- where
--
--   id-pkix OBJECT IDENTIFIER ::= {
--      iso(1) identified-organization(3)
--      dod(6) internet(1) security(5) mechanisms(5) pkix(7)}
-- and
--   id-it   OBJECT IDENTIFIER ::= {id-pkix 4}
--
--
-- This construct MAY also be used to define new PKIX Certificate
-- Management Protocol request and response messages or general-
-- purpose (e.g., announcement) messages for future needs or for
-- specific environments.

GenMsgContent ::= SEQUENCE OF InfoTypeAndValue

-- May be sent by EE, RA, or CA (depending on message content).
-- The OPTIONAL infoValue parameter of InfoTypeAndValue will
-- typically be omitted for some of the examples given above.
-- The receiver is free to ignore any contained OIDs that it
-- does not recognize.  If sent from EE to CA, the empty set
-- indicates that the CA may send
-- any/all information that it wishes.

GenRepContent ::= SEQUENCE OF InfoTypeAndValue
-- The receiver MAY ignore any contained OIDs that it does not
-- recognize.

ErrorMsgContent ::= SEQUENCE {
    pKIStatusInfo          PKIStatusInfo,
    errorCode              INTEGER           OPTIONAL,
    -- implementation-specific error codes
    errorDetails           PKIFreeText       OPTIONAL
    -- implementation-specific error details
}
```

```
PollReqContent ::= SEQUENCE OF SEQUENCE {
    certReqId              INTEGER
}

PollRepContent ::= SEQUENCE OF SEQUENCE {
    certReqId              INTEGER,
    checkAfter             INTEGER,  -- time in seconds
    reason                 PKIFreeText OPTIONAL
}

--
-- Extended key usage extension for PKI entities used in CMP
-- operations, added due to the changes made in
-- CMP Updates [RFC9480]
-- The EKUs for the CA and RA are reused from CMC, as defined in
-- [RFC6402]
--

-- id-kp-cmcCA OBJECT IDENTIFIER ::= { id-kp 27 }
-- id-kp-cmcRA OBJECT IDENTIFIER ::= { id-kp 28 }
id-kp-cmKGA OBJECT IDENTIFIER ::= { id-kp 32 }

-- There is no 1988 ASN.1 module of PKCS #9 available to import the
-- syntax of the localKeyId attribute type and value from. Therefore,
-- the syntax is added here as needed for the updates made in
-- CMP Updates [RFC9480].

pkcs-9 OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840)
                              rsadsi(113549) pkcs(1) 9}

pkcs-9-at-localKeyId OBJECT IDENTIFIER ::= {pkcs-9 21}

LocalKeyIdValue ::= OCTET STRING

END -- of CMP module
```

A.2. Update to RFC 5912 - 2002 ASN.1 Module

This section contains the updated 2002 ASN.1 module for [RFC5912]. This module replaces the module in Section 9 of [RFC5912]. The module contains those changes to the normative ASN.1 module from Appendix F of [RFC4210] that were to update to the 2002 ASN.1 standard done in [RFC5912], as well as changes made in this document.

```
PKIXCMP-2021
    { iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7) id-mod(0)
    id-mod-cmp2021-02(100) }
DEFINITIONS EXPLICIT TAGS ::=
BEGIN
IMPORTS

AttributeSet{}, SingleAttribute{}, Extensions{}, EXTENSION, ATTRIBUTE
FROM PKIX-CommonTypes-2009
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-pkixCommon-02(57)}
```

```
AlgorithmIdentifier{}, SIGNATURE-ALGORITHM, ALGORITHM,
    DIGEST-ALGORITHM, MAC-ALGORITHM
FROM AlgorithmInformation-2009
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0)
    id-mod-algorithmInformation-02(58)}

Certificate, CertificateList, Time, id-kp
FROM PKIX1Explicit-2009
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-explicit-02(51)}

DistributionPointName, GeneralNames, GeneralName, KeyIdentifier
FROM PKIX1Implicit-2009
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-implicit-02(59)}

CertTemplate, PKIPublicationInfo, EncryptedKey, CertId,
    CertReqMessages, Controls, RegControlSet, id-regCtrl
FROM PKIXCRMF-2009
    { iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7) id-mod(0)
    id-mod-crmf2005-02(55) }
    -- The import of EncryptedKey is added due to the updates made
    -- in CMP Updates [RFC9480]. EncryptedValue does not need to
    -- be imported anymore and is therefore removed here.

-- See also the behavioral clarifications to CRMF codified in
-- Appendix C of this specification.

CertificationRequest
FROM PKCS-10
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-pkcs10-2009(69)}
-- (specified in [RFC2986] with 1993 ASN.1 syntax and IMPLICIT
-- tags).  Alternatively, implementers may directly include
-- the syntax of [RFC2986] in this module.

localKeyId
FROM PKCS-9
    {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    modules(0) pkcs-9(1)}
    -- The import of localKeyId is added due to the updates made in
    -- CMP Updates [RFC9480].

EnvelopedData, SignedData
FROM CryptographicMessageSyntax-2009
    {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) modules(0) id-mod-cms-2004-02(41)}
    -- The import of EnvelopedData and SignedData is added due to
    -- the updates made in CMP Updates [RFC9480].
;

-- The rest of the module contains locally defined OIDs and
-- constructs:

CMPCertificate ::= CHOICE { x509v3PKCert Certificate, ... }
```

```
-- This syntax, while bits-on-the-wire compatible with the
-- standard X.509 definition of "Certificate", allows the
-- possibility of future certificate types (such as X.509
-- attribute certificates, card-verifiable
-- certificates, or other kinds of certificates) within this
-- Certificate Management Protocol, should a need ever arise to
-- support such generality.  Those implementations that do not
-- foresee a need to ever support other certificate types MAY, if
-- they wish, comment out the above structure and "uncomment" the
-- following one prior to compiling this ASN.1 module.  (Note that
-- interoperability with implementations that don't do this will be
-- unaffected by this change.)

-- CMPCertificate ::= Certificate

PKIMessage ::= SEQUENCE {
    header           PKIHeader,
    body             PKIBody,
    protection   [0] PKIProtection OPTIONAL,
    extraCerts   [1] SEQUENCE SIZE (1..MAX) OF CMPCertificate
                     OPTIONAL }

PKIMessages ::= SEQUENCE SIZE (1..MAX) OF PKIMessage

PKIHeader ::= SEQUENCE {
    pvno                INTEGER     { cmp1999(1), cmp2000(2),
                                      cmp2012(3) },
    sender              GeneralName,
    -- identifies the sender
    recipient           GeneralName,
    -- identifies the intended recipient
    messageTime     [0] GeneralizedTime         OPTIONAL,
    -- time of production of this message (used when the sender
    -- believes that the transport will be "suitable", i.e.,
    -- that the time will still be meaningful upon receipt)
    protectionAlg   [1] AlgorithmIdentifier{ALGORITHM, {...}}
                            OPTIONAL,
    -- algorithm used for the calculation of protection bits
    senderKID       [2] KeyIdentifier           OPTIONAL,
    recipKID        [3] KeyIdentifier           OPTIONAL,
    -- to identify specific keys used for protection
    transactionID   [4] OCTET STRING            OPTIONAL,
    -- identifies the transaction, i.e., this will be the same in
    -- corresponding request, response, certConf, and PKIConf
    -- messages
    senderNonce     [5] OCTET STRING            OPTIONAL,
    recipNonce      [6] OCTET STRING            OPTIONAL,
    -- nonces used to provide replay protection, senderNonce
    -- is inserted by the creator of this message; recipNonce
    -- is a nonce previously inserted in a related message by
    -- the intended recipient of this message.
    freeText        [7] PKIFreeText             OPTIONAL,
    -- this may be used to indicate context-specific instructions
    -- (this field is intended for human consumption)
    generalInfo     [8] SEQUENCE SIZE (1..MAX) OF
                        InfoTypeAndValue     OPTIONAL
    -- this may be used to convey context-specific information
    -- (this field not primarily intended for human consumption)
```

```
}

PKIFreeText ::= SEQUENCE SIZE (1..MAX) OF UTF8String
    -- text encoded as a UTF-8 string [RFC3629]

PKIBody ::= CHOICE {       -- message-specific body elements
    ir       [0]  CertReqMessages,        --Initialization Request
    ip       [1]  CertRepMessage,         --Initialization Response
    cr       [2]  CertReqMessages,        --Certification Request
    cp       [3]  CertRepMessage,         --Certification Response
    p10cr    [4]  CertificationRequest,   --imported from [RFC2986]
    popdecc  [5]  POPODecKeyChallContent, --pop Challenge
    popdecr  [6]  POPODecKeyRespContent,  --pop Response
    kur      [7]  CertReqMessages,        --Key Update Request
    kup      [8]  CertRepMessage,         --Key Update Response
    krr      [9]  CertReqMessages,        --Key Recovery Request
    krp      [10] KeyRecRepContent,       --Key Recovery Response
    rr       [11] RevReqContent,          --Revocation Request
    rp       [12] RevRepContent,          --Revocation Response
    ccr      [13] CertReqMessages,        --Cross-Cert. Request
    ccp      [14] CertRepMessage,         --Cross-Cert. Response
    ckuann   [15] CAKeyUpdAnnContent,     --CA Key Update Ann.
    cann     [16] CertAnnContent,         --Certificate Ann.
    rann     [17] RevAnnContent,          --Revocation Ann.
    crlann   [18] CRLAnnContent,          --CRL Announcement
    pkiconf  [19] PKIConfirmContent,      --Confirmation
    nested   [20] NestedMessageContent,   --Nested Message
    genm     [21] GenMsgContent,          --General Message
    genp     [22] GenRepContent,          --General Response
    error    [23] ErrorMsgContent,        --Error Message
    certConf [24] CertConfirmContent,     --Certificate Confirm
    pollReq  [25] PollReqContent,         --Polling Request
    pollRep  [26] PollRepContent          --Polling Response
}

PKIProtection ::= BIT STRING

ProtectedPart ::= SEQUENCE {
    header    PKIHeader,
    body      PKIBody }

id-PasswordBasedMac OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    usa(840) nt(113533) nsn(7) algorithms(66) 13 }
PBMParameter ::= SEQUENCE {
    salt                OCTET STRING,
    -- Note: Implementations MAY wish to limit acceptable sizes
    -- of this string to values appropriate for their environment
    -- in order to reduce the risk of denial-of-service attacks.
    owf                 AlgorithmIdentifier{DIGEST-ALGORITHM, {...}},
    -- AlgId for a One-Way Function
    iterationCount      INTEGER,
    -- number of times the OWF is applied
    -- Note: Implementations MAY wish to limit acceptable sizes
    -- of this integer to values appropriate for their environment
    -- in order to reduce the risk of denial-of-service attacks.
    mac                 AlgorithmIdentifier{MAC-ALGORITHM, {...}}
    -- the MAC AlgId (e.g., HMAC-SHA256, AES-GMAC [RFC9481],
    -- or HMAC [RFC2104, RFC2202])
```

```
}

id-DHBasedMac OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    usa(840) nt(113533) nsn(7) algorithms(66) 30 }
DHBMParameter ::= SEQUENCE {
    owf                 AlgorithmIdentifier{DIGEST-ALGORITHM, {...}},
    -- AlgId for a One-Way Function
    mac                 AlgorithmIdentifier{MAC-ALGORITHM, {...}}
    -- the MAC AlgId (e.g., HMAC-SHA256, AES-GMAC [RFC9481],
    -- or HMAC [RFC2104, RFC2202])
}

PKIStatus ::= INTEGER {
    accepted              (0),
    -- you got exactly what you asked for
    grantedWithMods       (1),
    -- you got something like what you asked for; the
    -- requester is responsible for ascertaining the differences
    rejection             (2),
    -- you don't get it, more information elsewhere in the message
    waiting               (3),
    -- the request body part has not yet been processed; expect to
    -- hear more later (note: proper handling of this status
    -- response MAY use the polling req/rep PKIMessages specified
    -- in Section 5.3.22 of [RFC4210]; alternatively, polling in the
    -- underlying transport layer MAY have some utility in this
    -- regard)
    revocationWarning     (4),
    -- this message contains a warning that a revocation is
    -- imminent
    revocationNotification (5),
    -- notification that a revocation has occurred
    keyUpdateWarning      (6)
    -- update already done for the oldCertId specified in
    -- CertReqMsg
}

PKIFailureInfo ::= BIT STRING {
-- since we can fail in more than one way!
-- More codes may be added in the future if/when required.
    badAlg              (0),
    -- unrecognized or unsupported algorithm identifier
    badMessageCheck     (1),
    -- integrity check failed (e.g., signature did not verify)
    badRequest          (2),
    -- transaction not permitted or supported
    badTime             (3),
    -- messageTime was not sufficiently close to the system time,
    -- as defined by local policy
    badCertId           (4),
    -- no certificate could be found matching the provided criteria
    badDataFormat       (5),
    -- the data submitted has the wrong format
    wrongAuthority      (6),
    -- the authority indicated in the request is different from the
    -- one creating the response token
    incorrectData       (7),
    -- the requester's data is incorrect (for notary services)
```

```
    missingTimeStamp    (8),
    -- when the timestamp is missing but should be there
    -- (by policy)
    badPOP              (9),
    -- the proof-of-possession failed
    certRevoked         (10),
    -- the certificate has already been revoked
    certConfirmed       (11),
    -- the certificate has already been confirmed
    wrongIntegrity      (12),
    -- not valid integrity, based on the password instead of the
    -- signature or vice versa
    badRecipientNonce   (13),
    -- not valid recipient nonce, either missing or wrong value
    timeNotAvailable    (14),
    -- the TSA's time source is not available
    unacceptedPolicy    (15),
    -- the requested TSA policy is not supported by the TSA
    unacceptedExtension (16),
    -- the requested extension is not supported by the TSA
    addInfoNotAvailable (17),
    -- the additional information requested could not be
    -- understood or is not available
    badSenderNonce      (18),
    -- not valid sender nonce, either missing or wrong size
    badCertTemplate     (19),
    -- not valid cert. template or missing mandatory information
    signerNotTrusted    (20),
    -- signer of the message unknown or not trusted
    transactionIdInUse  (21),
    -- the transaction identifier is already in use
    unsupportedVersion  (22),
    -- the version of the message is not supported
    notAuthorized       (23),
    -- the sender was not authorized to make the preceding
    -- request or perform the preceding action
    systemUnavail       (24),
    -- the request cannot be handled due to system unavailability
    systemFailure       (25),
    -- the request cannot be handled due to system failure
    duplicateCertReq    (26)
    -- the certificate cannot be issued because a duplicate
    -- certificate already exists
}

PKIStatusInfo ::= SEQUENCE {
    status        PKIStatus,
    statusString  PKIFreeText     OPTIONAL,
    failInfo      PKIFailureInfo  OPTIONAL }

OOBCert ::= CMPCertificate

OOBCertHash ::= SEQUENCE {
    hashAlg     [0] AlgorithmIdentifier{DIGEST-ALGORITHM, {...}}
                    OPTIONAL,
    certId      [1] CertId                  OPTIONAL,
    hashVal         BIT STRING
    -- hashVal is calculated over the DER encoding of the
```

```
    -- self-signed certificate with the identifier certID.
}

POPODecKeyChallContent ::= SEQUENCE OF Challenge
-- One Challenge per encryption key certification request (in the
-- same order as these requests appear in CertReqMessages)

Challenge ::= SEQUENCE {
    owf                 AlgorithmIdentifier{DIGEST-ALGORITHM, {...}}
                            OPTIONAL,
    -- MUST be present in the first Challenge; MAY be omitted in
    -- any subsequent Challenge in POPODecKeyChallContent (if
    -- omitted, then the owf used in the immediately preceding
    -- Challenge is to be used)
    witness             OCTET STRING,
    -- the result of applying the One-Way Function (owf) to a
    -- randomly generated INTEGER, A (Note that a different
    -- INTEGER MUST be used for each Challenge.)
    challenge           OCTET STRING
    -- the encryption (under the public key for which the cert.
    -- request is being made) of Rand
}

-- Rand was added in CMP Updates [RFC9480]

Rand ::= SEQUENCE {
-- Rand is encrypted under the public key to form the challenge
-- in POPODecKeyChallContent
   int                  INTEGER,
   -- the randomly generated INTEGER A (above)
   sender               GeneralName
   -- the sender's name (as included in PKIHeader)
}

POPODecKeyRespContent ::= SEQUENCE OF INTEGER
-- One INTEGER per encryption key certification request (in the
-- same order as these requests appear in CertReqMessages).  The
-- retrieved INTEGER A (above) is returned to the sender of the
-- corresponding Challenge.

CertRepMessage ::= SEQUENCE {
    caPubs       [1] SEQUENCE SIZE (1..MAX) OF CMPCertificate
                  OPTIONAL,
    response         SEQUENCE OF CertResponse }

CertResponse ::= SEQUENCE {
    certReqId           INTEGER,
    -- to match this response with the corresponding request (a value
    -- of -1 is to be used if certReqId is not specified in the
    -- corresponding request, which can only be a p10cr)
    status              PKIStatusInfo,
    certifiedKeyPair    CertifiedKeyPair    OPTIONAL,
    rspInfo             OCTET STRING        OPTIONAL
    -- analogous to the id-regInfo-utf8Pairs string defined
    -- for regInfo in CertReqMsg [RFC4211]
}

CertifiedKeyPair ::= SEQUENCE {
```

```
    certOrEncCert       CertOrEncCert,
    privateKey      [0] EncryptedKey      OPTIONAL,
    -- See [RFC4211] for comments on encoding.
    -- Changed from Encrypted Value to EncryptedKey as a CHOICE of
    -- EncryptedValue and EnvelopedData due to the changes made in
    -- CMP Updates [RFC9480].
    -- Using the choice EncryptedValue is bit-compatible to the
    -- syntax without this change.
    publicationInfo [1] PKIPublicationInfo  OPTIONAL }

CertOrEncCert ::= CHOICE {
    certificate     [0] CMPCertificate,
    encryptedCert   [1] EncryptedKey
    -- Changed from Encrypted Value to EncryptedKey as a CHOICE of
    -- EncryptedValue and EnvelopedData due to the changes made in
    -- CMP Updates [RFC9480].
    -- Using the choice EncryptedValue is bit-compatible to the
    -- syntax without this change.
}

KeyRecRepContent ::= SEQUENCE {
    status                  PKIStatusInfo,
    newSigCert          [0] CMPCertificate OPTIONAL,
    caCerts             [1] SEQUENCE SIZE (1..MAX) OF
                                CMPCertificate OPTIONAL,
    keyPairHist         [2] SEQUENCE SIZE (1..MAX) OF
                                CertifiedKeyPair OPTIONAL }

RevReqContent ::= SEQUENCE OF RevDetails

RevDetails ::= SEQUENCE {
    certDetails         CertTemplate,
    -- allows the requester to specify as much as they can about
    -- the cert. for which revocation is requested
    -- (e.g., for cases in which serialNumber is not available)
    crlEntryDetails     Extensions{{...}}    OPTIONAL
    -- requested crlEntryExtensions
}

RevRepContent ::= SEQUENCE {
    status       SEQUENCE SIZE (1..MAX) OF PKIStatusInfo,
    -- in the same order as was sent in RevReqContent
    revCerts [0] SEQUENCE SIZE (1..MAX) OF CertId OPTIONAL,
    -- IDs for which revocation was requested
    -- (same order as status)
    crls     [1] SEQUENCE SIZE (1..MAX) OF CertificateList OPTIONAL
    -- the resulting CRLs (there may be more than one)
}

CAKeyUpdAnnContent ::= SEQUENCE {
    oldWithNew   CMPCertificate, -- old pub signed with new priv
    newWithOld   CMPCertificate, -- new pub signed with old priv
    newWithNew   CMPCertificate  -- new pub signed with new priv
}

CertAnnContent ::= CMPCertificate

RevAnnContent ::= SEQUENCE {
```

```
    status              PKIStatus,
    certId              CertId,
    willBeRevokedAt     GeneralizedTime,
    badSinceDate        GeneralizedTime,
    crlDetails          Extensions{{...}}  OPTIONAL
    -- extra CRL details (e.g., crl number, reason, location, etc.)
}

CRLAnnContent ::= SEQUENCE OF CertificateList
PKIConfirmContent ::= NULL

NestedMessageContent ::= PKIMessages

-- CertReqTemplateContent, AttributeTypeAndValue,
-- ExpandedRegControlSet, id-regCtrl-altCertTemplate,
-- AltCertTemplate, regCtrl-algId, id-regCtrl-algId, AlgIdCtrl,
-- regCtrl-rsaKeyLen, id-regCtrl-rsaKeyLen, and RsaKeyLenCtrl
-- were added in CMP Updates [RFC9480]

CertReqTemplateContent ::= SEQUENCE {
   certTemplate           CertTemplate,
   -- prefilled certTemplate structure elements
   -- The SubjectPublicKeyInfo field in the certTemplate MUST NOT
   -- be used.
   keySpec                Controls OPTIONAL
   -- MAY be used to specify supported algorithms
   -- Controls  ::= SEQUENCE SIZE (1..MAX) OF AttributeTypeAndValue
   -- as specified in CRMF [RFC4211]
   }

AttributeTypeAndValue ::= SingleAttribute{{ ... }}

ExpandedRegControlSet ATTRIBUTE ::= { RegControlSet |
   regCtrl-altCertTemplate | regCtrl-algId | regCtrl-rsaKeyLen, ... }

regCtrl-altCertTemplate ATTRIBUTE ::=
   { TYPE AltCertTemplate IDENTIFIED BY id-regCtrl-altCertTemplate }

id-regCtrl-altCertTemplate OBJECT IDENTIFIER ::= { id-regCtrl 7 }

AltCertTemplate ::= AttributeTypeAndValue
   -- specifies a template for a certificate other than an X.509v3
   -- public key certificate

regCtrl-algId ATTRIBUTE ::=
   { TYPE AlgIdCtrl IDENTIFIED BY id-regCtrl-algId }

id-regCtrl-algId OBJECT IDENTIFIER ::= { id-regCtrl 11 }

AlgIdCtrl ::= AlgorithmIdentifier{ALGORITHM, {...}}
   -- SHALL be used to specify supported algorithms other than RSA

regCtrl-rsaKeyLen ATTRIBUTE ::=
   { TYPE RsaKeyLenCtrl IDENTIFIED BY id-regCtrl-rsaKeyLen }

id-regCtrl-rsaKeyLen OBJECT IDENTIFIER ::= { id-regCtrl 12 }

RsaKeyLenCtrl ::= INTEGER (1..MAX)
```

```
   -- SHALL be used to specify supported RSA key lengths

-- RootCaKeyUpdateContent, CRLSource, and CRLStatus were added in
-- CMP Updates [RFC9480]

RootCaKeyUpdateContent ::= SEQUENCE {
   newWithNew       CMPCertificate,
   -- new root CA certificate
   newWithOld   [0] CMPCertificate OPTIONAL,
   -- X.509 certificate containing the new public root CA key
   -- signed with the old private root CA key
   oldWithNew   [1] CMPCertificate OPTIONAL
   -- X.509 certificate containing the old public root CA key
   -- signed with the new private root CA key
   }

CRLSource ::= CHOICE {
   dpn          [0] DistributionPointName,
   issuer       [1] GeneralNames }

CRLStatus ::= SEQUENCE {
   source       CRLSource,
   thisUpdate   Time OPTIONAL }

INFO-TYPE-AND-VALUE ::= TYPE-IDENTIFIER

InfoTypeAndValue ::= SEQUENCE {
    infoType    INFO-TYPE-AND-VALUE.
                    &id({SupportedInfoSet}),
    infoValue   INFO-TYPE-AND-VALUE.
                    &Type({SupportedInfoSet}{@infoType}) }

SupportedInfoSet INFO-TYPE-AND-VALUE ::= { ... }

-- Example InfoTypeAndValue contents include, but are not limited
-- to, the following (uncomment in this ASN.1 module and use as
-- appropriate for a given environment):
--
--   id-it-caProtEncCert    OBJECT IDENTIFIER ::= {id-it 1}
--      CAProtEncCertValue      ::= CMPCertificate
--   id-it-signKeyPairTypes OBJECT IDENTIFIER ::= {id-it 2}
--      SignKeyPairTypesValue   ::= SEQUENCE SIZE (1..MAX) OF
--                                      AlgorithmIdentifier{{...}}
--   id-it-encKeyPairTypes  OBJECT IDENTIFIER ::= {id-it 3}
--      EncKeyPairTypesValue    ::= SEQUENCE SIZE (1..MAX) OF
--                                      AlgorithmIdentifier{{...}}
--   id-it-preferredSymmAlg OBJECT IDENTIFIER ::= {id-it 4}
--      PreferredSymmAlgValue   ::= AlgorithmIdentifier{{...}}
--   id-it-caKeyUpdateInfo  OBJECT IDENTIFIER ::= {id-it 5}
--      CAKeyUpdateInfoValue    ::= CAKeyUpdAnnContent
--   id-it-currentCRL       OBJECT IDENTIFIER ::= {id-it 6}
--      CurrentCRLValue         ::= CertificateList
--   id-it-unsupportedOIDs  OBJECT IDENTIFIER ::= {id-it 7}
--      UnsupportedOIDsValue    ::= SEQUENCE SIZE (1..MAX) OF
--                                          OBJECT IDENTIFIER
--   id-it-keyPairParamReq  OBJECT IDENTIFIER ::= {id-it 10}
--      KeyPairParamReqValue    ::= OBJECT IDENTIFIER
--   id-it-keyPairParamRep  OBJECT IDENTIFIER ::= {id-it 11}
```

```
--      KeyPairParamRepValue    ::= AlgorithmIdentifier{{...}}
--   id-it-revPassphrase    OBJECT IDENTIFIER ::= {id-it 12}
--      RevPassphraseValue      ::= EncryptedKey
--      - Changed from Encrypted Value to EncryptedKey as a CHOICE
--      - of EncryptedValue and EnvelopedData due to the changes
--      - made in CMP Updates [RFC9480]
--      - Using the choice EncryptedValue is bit-compatible to
--      - the syntax without this change
--   id-it-implicitConfirm  OBJECT IDENTIFIER ::= {id-it 13}
--      ImplicitConfirmValue    ::= NULL
--   id-it-confirmWaitTime  OBJECT IDENTIFIER ::= {id-it 14}
--      ConfirmWaitTimeValue    ::= GeneralizedTime
--   id-it-origPKIMessage   OBJECT IDENTIFIER ::= {id-it 15}
--      OrigPKIMessageValue     ::= PKIMessages
--   id-it-suppLangTags     OBJECT IDENTIFIER ::= {id-it 16}
--      SuppLangTagsValue       ::= SEQUENCE OF UTF8String
--   id-it-caCerts          OBJECT IDENTIFIER ::= {id-it 17}
--      CaCertsValue            ::= SEQUENCE SIZE (1..MAX) OF
--                                      CMPCertificate
--      - id-it-caCerts added in CMP Updates [RFC9480]
--   id-it-rootCaKeyUpdate  OBJECT IDENTIFIER ::= {id-it 18}
--      RootCaKeyUpdateValue    ::= RootCaKeyUpdateContent
--      - id-it-rootCaKeyUpdate added in CMP Updates [RFC9480]
--   id-it-certReqTemplate  OBJECT IDENTIFIER ::= {id-it 19}
--      CertReqTemplateValue    ::= CertReqTemplateContent
--      - id-it-certReqTemplate added in CMP Updates [RFC9480]
--   id-it-rootCaCert       OBJECT IDENTIFIER ::= {id-it 20}
--      RootCaCertValue         ::= CMPCertificate
--      - id-it-rootCaCert added in CMP Updates [RFC9480]
--   id-it-certProfile      OBJECT IDENTIFIER ::= {id-it 21}
--      CertProfileValue        ::= SEQUENCE SIZE (1..MAX) OF
--                                                  UTF8String
--      - id-it-certProfile added in CMP Updates [RFC9480]
--   id-it-crlStatusList    OBJECT IDENTIFIER ::= {id-it 22}
--   CRLStatusListValue         ::= SEQUENCE SIZE (1..MAX) OF
--                                                   CRLStatus
--      - id-it-crlStatusList added in CMP Updates [RFC9480]
--   id-it-crls             OBJECT IDENTIFIER ::= {id-it 23}
--   CRLsValue                  ::= SEQUENCE SIZE (1..MAX) OF
--                                                 CertificateList
--      - id-it-crls added in CMP Updates [RFC9480]
--
-- where
--
--   id-pkix OBJECT IDENTIFIER ::= {
--      iso(1) identified-organization(3)
--      dod(6) internet(1) security(5) mechanisms(5) pkix(7)}
-- and
--   id-it   OBJECT IDENTIFIER ::= {id-pkix 4}
--
--
-- This construct MAY also be used to define new PKIX Certificate
-- Management Protocol request and response messages or general-
-- purpose (e.g., announcement) messages for future needs or for
-- specific environments.

GenMsgContent ::= SEQUENCE OF InfoTypeAndValue
```

```
-- May be sent by EE, RA, or CA (depending on message content).
-- The OPTIONAL infoValue parameter of InfoTypeAndValue will
-- typically be omitted for some of the examples given above.
-- The receiver is free to ignore any contained OIDs that it
-- does not recognize.  If sent from EE to CA, the empty set
-- indicates that the CA may send
-- any/all information that it wishes.

GenRepContent ::= SEQUENCE OF InfoTypeAndValue
-- The receiver MAY ignore any contained OIDs that it does not
-- recognize.

ErrorMsgContent ::= SEQUENCE {
    pKIStatusInfo          PKIStatusInfo,
    errorCode              INTEGER           OPTIONAL,
    -- implementation-specific error codes
    errorDetails           PKIFreeText       OPTIONAL
    -- implementation-specific error details
}

CertConfirmContent ::= SEQUENCE OF CertStatus

CertStatus ::= SEQUENCE {
    certHash    OCTET STRING,
    -- the hash of the certificate, using the same hash algorithm
    -- as is used to create and verify the certificate signature
    certReqId   INTEGER,
    -- to match this confirmation with the corresponding req/rep
    statusInfo  PKIStatusInfo OPTIONAL,
    hashAlg [0] AlgorithmIdentifier{DIGEST-ALGORITHM, {...}} OPTIONAL
    -- the hash algorithm to use for calculating certHash
    -- SHOULD NOT be used in all cases where the AlgorithmIdentifier
    -- of the certificate signature specifies a hash algorithm
   }

PollReqContent ::= SEQUENCE OF SEQUENCE {
    certReqId              INTEGER }

PollRepContent ::= SEQUENCE OF SEQUENCE {
    certReqId              INTEGER,
    checkAfter             INTEGER,  -- time in seconds
    reason                 PKIFreeText OPTIONAL }

--
-- Extended key usage extension for PKI entities used in CMP
-- operations, added due to the changes made in
-- CMP Updates [RFC9480]
-- The EKUs for the CA and RA are reused from CMC, as defined in
-- [RFC6402]
--

-- id-kp-cmcCA OBJECT IDENTIFIER ::= { id-kp 27 }
-- id-kp-cmcRA OBJECT IDENTIFIER ::= { id-kp 28 }
id-kp-cmKGA OBJECT IDENTIFIER ::= { id-kp 32 }

END
```

Acknowledgements

Special thanks goes to Jim Schaad for his guidance and the inspiration to structure and write this document like [RFC6402], which updates CMC. Special thanks also goes to Russ Housley, Lijun Liao, Martin Peylo, and Tomas Gustavsson for reviewing and providing valuable suggestions on improving this document.

We also thank all reviewers of this document for their valuable feedback.

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

John Gray
Entrust
1187 Park Place
Minneapolis, MN 55379
United States of America
Email: john.gray@entrust.com
URI: https://www.entrust.com