lamps
Internet-Draft
Intended status: Informational
Expires: 4 May 2021

D.K. Gillmor
ACLU

31 October 2020

Guidance on End-to-End E-mail Security
draft-dkg-lamps-e2e-mail-guidance-00

Abstract

End-to-end cryptographic protections for e-mail messages can provide useful security. However, the standards for providing cryptographic protection are extremely flexible. That flexibility can trap users and cause surprising failures. This document offers guidance for mail user agent implementers that need to compose or interpret e-mail messages with end-to-end cryptographic protection. It provides a useful set of vocabulary as well as suggestions to avoid common failures.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 4 May 2021.

Table of Contents

1. Introduction

E-mail end-to-end security using S/MIME [RFC8551] and PGP/MIME [RFC3156] cryptographic standards can provide integrity, authentication and confidentiality to MIME e-mail messages [RFC4289].

However, there are many ways that a receiving mail user agent can misinterpret or accidentally break these security guarantees (e.g., [EFAIL]).

A mail user agent that interprets a message with end-to-end cryptographic protections needs to do so defensively, staying alert to different ways that these protections can be bypassed by mangling (either malicious or accidental) or a failed user experience.

A mail user agent that generates a message with end-to-end cryptographic protections should be aware of these defensive interpretation strategies, and should compose any new outbound message conservatively if they want the protections to remain intact.

1.1. Requirements Language

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] and [RFC8174]) when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

For the purposes of this document, we define the following concepts:

* _MUA_ is short for Mail User Agent; an e-mail client.

* _Protection_ of message data refers to cryptographic encryption and/or signatures, providing confidentiality, authenticity, and/or integrity.

* _Cryptographic Layer_, _Cryptographic Envelope_, _Cryptographic Payload_, and _Errant Cryptographic Layer_ are defined in Section 4

* A _well-formed_ e-mail message with cryptographic protection has both a _Cryptographic Envelope_ and a _Cryptographic Payload_,

* _Structural Headers_ are documented in Section 1.2.1.

1.2.1. Structural Headers

A message header whose name begins with "Content-" is referred to in this document as a "structural" header.

These headers indicate something about the specific MIME part they are attached to, and cannot be transferred or copied to other parts without endangering the readability of the message.

This includes (but is not limited to):

* "Content-Type"

* "Content-Transfer-Encoding"

* "Content-Disposition"

FIXME: are there any non-"Content-*" headers we should consider as structural?

2. Usability

The end user (the operator of the MUA) is unlikely to understand complex end-to-end cryptographic protections on any e-mail message, so keep it simple.

For clarity to the user, any cryptographic protections should apply to the message as a whole, not just to some subparts.

This is true for message composition: the standard message composition user interface of an MUA should offer minimal controls which indicate which types of protection to apply to the new message as a whole.

This is also true for message interpretation: the standard message rendering user interface of an MUA should offer a minimal, clear indicator about the end-to-end cryptographic status of the message as a whole.

3. Types of Protection

A given message might be:

* signed,

* encrypted, or

* both signed and encrypted.

Given that many e-mail messages offer no cryptographic protections, the user needs to be able to detect which protections are present for any given message.

4. Cryptographic MIME Message Structure

Implementations use the structure of an e-mail message to protect the headers. This section establishes some conventions about how to think about message structure.

4.1. Cryptographic Layers

"Cryptographic Layer" refers to a MIME substructure that supplies some cryptographic protections to an internal MIME subtree. The internal subtree is known as the "protected part" though of course it may itself be a multipart object.

In the diagrams below, "" (DOWNWARDS ARROW FROM BAR, U+21A7) indicates "decrypts to", and "" (DOWNWARDS WHITE ARROW, U+21E9) indicates "unwraps to".

4.1.1. S/MIME Cryptographic Layers

For S/MIME [RFC8551], there are four forms of Cryptographic Layers: multipart/signed, PKCS#7 signed-data, PKCS7 enveloped-data, PKCS7 authEnveloped-data.

4.1.1.1. S/MIME Multipart Signed Cryptographic Layer

```
multipart/signed; protocol="application/pkcs7-signature"
 [protected part]
 application/pkcs7-signature
```

This MIME layer offers authentication and integrity.

4.1.1.2. S/MIME PKCS7 signed-data Cryptographic Layer

```
application/pkcs7-mime; smime-type="signed-data"
  (unwraps to)
 [protected part]
```

This MIME layer offers authentication and integrity.

4.1.1.3. S/MIME PKCS7 enveloped-data Cryptographic Layer

```
application/pkcs7-mime; smime-type="enveloped-data"
  (decrypts to)
 [protected part]
```

This MIME layer offers confidentiality.

4.1.1.4. S/MIME PKCS7 authEnveloped-data Cryptographic Layer

```
application/pkcs7-mime; smime-type="authEnveloped-data"
  (decrypts to)
 [protected part]
```

This MIME layer offers confidentiality and integrity.

Note that "enveloped-data" (Section 4.1.1.3) and "authEnveloped-data" (Section 4.1.1.4) have identical message structure and semantics. The only difference between the two is ciphertext malleability.

The examples in this document only include "enveloped-data", but the implications for that layer apply to "authEnveloped-data" as well.

4.1.1.5. PKCS7 Compression is NOT a Cryptographic Layer

The Cryptographic Message Syntax (CMS) provides a MIME compression layer ("smime-type="compressed-data""), as defined in [RFC3274]. While the compression layer is technically a part of CMS, it is not considered a Cryptographic Layer for the purposes of this document.

4.1.2. PGP/MIME Cryptographic Layers

For PGP/MIME [RFC3156] there are two forms of Cryptographic Layers, signing and encryption.

4.1.2.1. PGP/MIME Signing Cryptographic Layer (multipart/signed)

```
multipart/signed; protocol="application/pgp-signature"
 [protected part]
 application/pgp-signature
```

This MIME layer offers authenticity and integrity.

4.1.2.2. PGP/MIME Encryption Cryptographic Layer (multipart/encrypted)

```
multipart/encrypted
 application/pgp-encrypted
 application/octet-stream
   (decrypts to)
  [protected part]
```

Note that for PGP/MIME, this MIME layer can offer any of:

* confidentiality (via a Symmetrically Encrypted Data Packet, see Section 5.7 of [RFC4880]; a MUA MUST NOT generate this form due to ciphertext malleability)

* confidentiality and integrity (via a Symmetrically Encrypted Integrity Protected Data Packet (SEIPD), see section 5.13 of [RFC4880]), or

* confidentiality, integrity, and authenticity all together (by including an OpenPGP Signature Packet within the SEIPD).

4.2. Cryptographic Envelope

The Cryptographic Envelope is the largest contiguous set of Cryptographic Layers of an e-mail message starting with the outermost MIME type (that is, with the Content-Type of the message itself).

If the Content-Type of the message itself is not a Cryptographic Layer, then the message has no cryptographic envelope.

"Contiguous" in the definition above indicates that if a Cryptographic Layer is the protected part of another Cryptographic Layer, the layers together comprise a single Cryptographic Envelope.

Note that if a non-Cryptographic Layer intervenes, all Cryptographic Layers within the non-Cryptographic Layer _are not_ part of the Cryptographic Envelope. They are Errant Cryptographic Layers (see Section 4.5).

Note also that the ordering of the Cryptographic Layers implies different cryptographic properties. A signed-then-encrypted message is different than an encrypted-then-signed message. See Section 5.2.

4.3. Cryptographic Payload

The Cryptographic Payload of a message is the first non-Cryptographic Layer - the "protected part" - within the Cryptographic Envelope.

4.4. Types of Cryptographic Envelope

4.4.1. Simple Cryptographic Envelopes

As described above, if the "protected part" identified in the sction above is not itself a Cryptographic Layer, that part _is_ the Cryptographic Payload.

If the application wants to generate a message that is both encrypted and signed, it MAY use the simple MIME structure from Section 4.1.2.2 by ensuring that the [RFC4880] Encrypted Message within the "application/octet-stream" part contains an [RFC4880] Signed Message (the final option described in Section 4.1.2.2.

4.4.2. Multilayer Cryptographic Envelopes

It is possible to construct a Cryptographic Envelope consisting of multiple layers with either S/MIME or PGP/MIME , for example using the following structure:

```
A application/pkcs7-mime; smime-type="enveloped-data"
B   (decrypts to)
C  application/pkcs7-mime; smime-type="signed-data"
D    (unwraps to)
E   [protected part]
```

When handling such a message, the properties of the Cryptographic Envelope are derived from the series "A", "C".

As noted in Section 4.4.1, PGP/MIME applications also have a simpler MIME construction available with the same cryptographic properties.

4.5. Errant Crytptographic Layers

Due to confusion, malice, or well-intentioned tampering, a message may contain a Cryptographic Layer that is not part of the Cryptographic Envelope. Such a layer is an Errant Cryptographic Layer.

An Errant Cryptographic Layer SHOULD NOT contribute to the message's overall cryptographic state.

Guidance for dealing with Errant Cryptographic Layers can be found in Section 6.2.

4.5.1. Mailing List Wrapping

Some mailing list software will re-wrap a well-formed signed message before re-sending to add a footer, resulting in the following structure seen by recipients of the e-mail:

```
H multipart/mixed
I  multipart/signed
J  text/plain
K  application/pgp-signature
L  text/plain
```

In this message, "L" is the footer added by the mailing list. "I" is now an Errant Cryptographic Layer.

Note that this message has no Cryptographic Envelope at all.

It is NOT RECOMMENDED to produce e-mail messages with this structure, because the data in part "L" may appear to the user as though it were part of "J", though they have different cryptographic properties. In particular, if the user believes that the message is signed, but cannot distinguish "L" from "J" then the author of "L" can effectively tamper with content of the signed message, breaking the user's expectation of integrity and authenticity.

4.5.2. A Baroque Example

Consider a message with the following overcomplicated structure:

```
M multipart/encrypted
N  application/pgp-encrypted
O  application/octet-stream
P    (decrypts to)
Q   multipart/signed
R    multipart/mixed
S    multipart/signed
T    text/plain
U    application/pgp-signature
V    text/plain
W    application/pgp-signature
```

The 3 Cryptographic Layers in such a message are rooted in parts "M", "Q", and "S". But the Cryptographic Envelope of the message consists only of the properties derived from the series "M", "Q". The Cryptographic Payload of the message is part "R". Part "S" is an Errant Cryptographic Layer.

Note that this message has both a Cryptographic Envelope _and_ an Errant Cryptographic Layer.

It is NOT RECOMMENDED to generate messages with such complicated structures. Even if a receiving MUA can parse this structure properly, it is nearly impossible to render in a way that the user can reason about the cryptographic properties of part "T" compared to part "V".

5. Message Composition

This section describes the ideal composition of an e-mail message with end-to-end cryptographic protection. A message composed with this form is most likely to achieve its end-to-end security goals.

5.1. Message Composition Algorithm

This section roughly describes the steps that a MUA should use to compose a cryptographically-protected message that has a proper cryptographic envelope and payload.

The message composition algorithm takes three parameters:

* "origbody": the traditional unprotected message body as a well-formed MIME tree (possibly just a single MIME leaf part). As a well-formed MIME tree, "origbody" already has structural headers present (see Section 1.2.1).

* "origheaders": the intended non-structural headers for the message, represented here as a table mapping from header names to header values.. For example, "origheaders['From']" refers to the value of the "From" header that the composing MUA would typically place on the message before sending it.

* "crypto": The series of cryptographic protections to apply (for example, "sign with the secret key corresponding to X.509 certificate X, then encrypt to X.509 certificates X and Y"). This is a routine that accepts a MIME tree as input (the Cryptographic Payload), wraps the input in the appropriate Cryptographic Envelope, and returns the resultant MIME tree as output.

The algorithm returns a MIME object that is ready to be injected into the mail system:

* Apply "crypto" to "origbody", yielding MIME tree "output"

* For header name "h" in "origheaders":

  - Set header "h" of "output" to "origheaders[h]"

* Return "output"

5.2. Encryption Outside, Signature Inside

Users expect any message that is both signed and encrypted to be signed _inside_ the encryption, and not the other way around.

Putting the signature inside the encryption has two advantages:

* The details of the signature remain confidential, visible only to the parties capable of decryption.

* Any mail transport agent that modifies the message is unlikely to be able to accidentally break the signature.

A MUA SHOULD NOT generate an encrypted and signed message where the only signature is outside the encryption.

5.3. Avoid Offering Encrypted-only Messages

When generating an e-mail, the user has options about what forms of end-to-end cryptographic protections to apply to it.

In some cases, offering any end-to-end cryptographic protection is harmful: it may confuse the recipient and offer no benefit.

In other cases, signing a message is useful (authenticity and integrity are desirable) but encryption is either impossible (for example, if the sender does not know how to encrypt to all recipients) or meaningless (for example, an e-mail message to a mailing list that is intended to be be published to a public archive).

In other cases, full end-to-end confidentiality, authenticity, and integrity are desirable.

It is unclear what the use case is for an e-mail message with end-to-end confidentiality but without authenticity or integrity.

A reasonable MUA will keep its message composition interface simple, so when presenting the user with a choice of cryptographic protection, it SHOULD offer no more than three choices:

* no end-to-end cryptographic protection

* signing-only

* signed and encrypted

5.4. Composing a Reply Message

When replying to a message, most MUAs compose an initial draft of the reply that contains quoted text from the original message. A responsible MUA will take precautions to avoid leaking the cleartext of an encrypted message in such a reply.

If the original message was end-to-end encrypted, the replying MUA MUST either:

* compose the reply with end-to-end encryption, or

* avoid including quoted text from the original message.

In general, MUAs SHOULD prefer the first option: to compose an encrypted reply. This is what users expect.

However, in some circumstances, the replying MUA cannot compose an encrypted reply. For example, the MUA might not have a valid, unexpired, encryption-capable certificate for all recipients. This can also happen during composition when a user adds a new recipient into the reply, or manually toggles the cryptographic protections to remove encryption.

In this circumstance, the composing MUA SHOULD strip the quoted text from the original message.

Note additional nuance about replies to malformed messages that contain encryption in Section 6.2.2.1.

6. Message Interpretation

Despite the best efforts of well-intentioned senders to create e-mail messages with well-formed end-to-end cryptographic protection, receiving MUAs will inevitably encounter some messages with malformed end-to-end cryptographic protection.

This section offers guidance on dealing with both well-formed and malformed messages containing Cryptographic Layers.

6.1. Rendering Well-formed Messages

A message is well-formed when it has a Cryptographic Envelope, a Cryptographic Payload, and no Errant Cryptographic Layers. Rendering a well-formed message is straightforward.

The receiving MUA should evaluate and summarize the cryptographic properties of the Cryptographic Envelope, and display that status to the user in a secure, strictly-controlled part of the UI. In particular, the part of the UI used to render the cryptographic summary of the message MUST NOT be spoofable, modifiable, or otherwise controllable by the received message itself.

Aside from this cryptographic summary, the message itself should be rendered as though the Cryptographic Payload is the body of the message. The Cryptographic Layers themselves SHOULD not be rendered otherwise.

6.2. Errant Cryptographic Layers

If an incoming message has any Errant Cryptographic Layers, the interpreting MUA SHOULD ignore those layers when rendering the cryptographic summary of the message to the user.

6.2.1. Errant Signing Layer

When rendering a message with an Errant Cryptographic Layer that provides authenticity and integrity (via signatures), the message should be rendered by replacing the Cryptographic layer with the part it encloses.

For example, a message with this structure:

```
A multipart/mixed
B  text/plain
C  multipart/signed
D  image/jpeg
E  application/pgp-signature
F  text/plain
```

Should be rendered identically to this:

```
A multipart/mixed
B  text/plain
D  image/jpeg
F  text/plain
```

In such a situation, an MUA SHOULD NOT indicate in the cryptographic summary that the message is signed.

6.2.1.1. Exception: Mailing List Footers

The use case described in Section 4.5.1 is common enough in some contexts, that a MUA MAY decide to handle it as a special exception.

If the MUA determines that the message comes from a mailing list (it has a "List-ID" header), and it has a structure that appends a footer to a signing-only Cryptographic Layer with a valid signature, such as:

```
H multipart/mixed
I  multipart/signed
J  [protected part, may be arbitrary MIME subtree]
K  application/{pgp,pkcs7}-signature
L  [footer, typically text/plain]
```

or:

```
H multipart/mixed
I  application/pkcs7-mime; smime-type="signed-data"
    (unwraps to)
J  [protected part, may be an arbitrary MIME subtree]
L  [footer, typically text/plain]
```

Then, the MUA MAY indicate to the user that this is a signed message that has been wrapped by the mailing list.

In this case, the MUA MUST distinguish the footer (part "L") from the protected part (part "J") when rendering any information about the signature.

One way to do this is to offer the user two different views of the message: the "mailing list" view, which hides any cryptographic summary but shows the footer:

```
Cryptographic Protections: none
H multipart/mixed
J  [protected part, may be arbitrary MIME subtree]
L  [footer, typically text/plain]
```

or the "sender's view", which shows the cryptographic summary but hides the footer:

```
Cryptographic Protections: signed [details from part I]
J [protected part, may be arbitrary MIME subtree]
```

6.2.2. Errant Encryption Layer

An MUA may encounter a message with an Errant Cryptographic Layer that offers confidentiality (encryption), and the MUA is capable of decrypting it.

The user wants to be able to see the contents of any message that they receive, so an MUA in this situation SHOULD decrypt the part.

In this case, though, the MUA MUST NOT indicate in the message's cryptographic summary that the message itself was encrypted. Such an indication could be taken to mean that other (non-encrypted) parts of the message arrived with cryptographic confidentiality.

6.2.2.1. Replying to a Message with an Errant Encryption Layer

Note that there is an asymmetry here between rendering and replying to a message with an Errant Encryption Layer.

When rendering, the MUA does not indicate that the message was encrypted, even if some subpart of it was decrypted for rendering.

But when composing a reply that contains quoted text from the decrypted subpart, the reply message SHOULD be marked for encryption, as noted in {#composing-reply}.

Alternately, if the reply message cannot be encrypted (or if the user elects to not encrypt the reply), the composed reply MUST NOT include any material from the decrypted subpart.

6.3. Forwarded Messages with Cryptographic Protection

An incoming e-mail message may include an attached forwarded message, typically as a MIME subpart with "Content-Type: message/rfc822" ([RFC5322]) or "Content-Type: message/global" ([RFC5355]).

Regardless of the cryptographic protections and structure of the incoming message, the internal forwarded message may have its own Cryptographic Envelope.

The Cryptographic Layers that are part of the Cryptographic Envelope of the forwarded message are not Errant Cryptographic Layers of the surrounding message - they are simply layers that apply to the forwarded message itself.

The rendering MUA MUST NOT conflate the cryptographic protections of the forwarded message with the cryptographic protections of the incoming message.

The rendering MUA MAY render a cryptograpic summary of the protections afforded to the forwarded message by its own Cryptographic Envelope, as long as that rendering is unambiguously tied to the forwarded message itself.

6.4. Signature failures

A cryptographic signature may fail in multiple ways. A receiving MUA that discovers a failed signature should treat the message as though the signature did not exist. This is similar to the standard guidance for about failed DKIM signatures (see section 6.1 of [RFC6376]).

A MUA SHOULD NOT render a message with a failed signature as more dangerous or more dubious than a comparable message without any signature at all.

A MUA that encounters an encrypted-and-signed message where the signature is invalid SHOULD treat the message the same way that it would treat a message that is encryption-only.

Some different ways that a signature may be invalid on a given message:

* the signature is not cryptographically valid (the math fails).

* the signature relies on suspect cryptographic primitives (e.g. over a legacy digest algorithm, or was made by a weak key, e.g., 1024-bit R.SA)

* the signature is made by a certificate which the receiving MUA does not have access to.

* the certificate that made the signature was revoked.

* the certificate that made the signature was expired at the time that the signature was made.

* the certificate that made the signature does not correspond to the author of the message.

* the signature indicates that it was made at a time much before or much after from the date of the message itself.

A valid signature must pass all these tests, but of course invalid signatures may be invalid in more than one of the ways listed above.

7. Common Pitfalls and Guidelines

This section highlights a few "pitfalls" and guidelines based on these discussions and lessons learned.

FIXME: some possible additional commentary on:

* indexing and search of encrypted messages

* managing access to cryptographic secret keys that require user interaction

* secure deletion

* inline PGP, ugh

8. IANA Considerations

MAYBE: provide an indicator in the IANA header registry for which headers are "structural" ? This is probably unnecessary.

9. Security Considerations

This entire document addresses security considerations about end-to-end cryptographic protections for e-mail messages.

10. Document Considerations

[RFC Editor: please remove this section before publication]

This document is currently edited as markdown. Minor editorial changes can be suggested via merge requests at https://gitlab.com/dkg/e2e-mail-guidance or by e-mail to the author. Please direct all significant commentary to the public IETF LAMPS mailing list: spasm@ietf.org

10.1. Document History

11. Acknowledgements

The set of constructs and recommendations in this document are derived from discussions with many different implementers, including Alexey Melnikov, Bernie Hoeneisen, Bjarni Runar Einarsson, David Bremner, Holger Krekel, Jameson Rollins, juga, Patrick Brunschwig, and Vincent Breitmoser.

12. References

12.1. 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>.

[RFC3156] Elkins, M., Del Torto, D., Levien, R., and T. Roessler, "MIME Security with OpenPGP", RFC 3156, DOI 10.17487/RFC3156, August 2001, <https://www.rfc-editor.org/info/rfc3156>.

[RFC4289] Freed, N. and J. Klensin, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", BCP 13, RFC 4289, DOI 10.17487/RFC4289, December 2005, <https://www.rfc-editor.org/info/rfc4289>.

[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>.

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

12.2. Informative References

[EFAIL] "EFAIL", n.d., <https://efail.de>.

[I-D.draft-bre-openpgp-samples-01] Einarsson, B., juga, j., and D. Gillmor, "OpenPGP Example Keys and Certificates", Work in Progress, Internet-Draft, draft-bre-openpgp-samples-01, 20 December 2019, <http://www.ietf.org/internet-drafts/draft-bre-openpgp-samples-01.txt>.

[I-D.draft-dkg-lamps-samples-02] Gillmor, D., "S/MIME Example Keys and Certificates", Work in Progress, Internet-Draft, draft-dkg-lamps-samples-02, 24 December 2019, <http://www.ietf.org/internet-drafts/draft-dkg-lamps-samples-02.txt>.

[RFC3274] Gutmann, P., "Compressed Data Content Type for Cryptographic Message Syntax (CMS)", RFC 3274, DOI 10.17487/RFC3274, June 2002, <https://www.rfc-editor.org/info/rfc3274>.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <https://www.rfc-editor.org/info/rfc4880>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[RFC5355] Stillman, M., Ed., Gopal, R., Guttman, E., Sengodan, S., and M. Holdrege, "Threats Introduced by Reliable Server Pooling (RSerPool) and Requirements for Security in Response to Threats", RFC 5355, DOI 10.17487/RFC5355, September 2008, <https://www.rfc-editor.org/info/rfc5355>.

[RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", STD 76, RFC 6376, DOI 10.17487/RFC6376, September 2011, <https://www.rfc-editor.org/info/rfc6376>.

Appendix A. Test Vectors

FIXME: This document should contain examples of well-formed and malformed messages using cryptographic key material and certificates from [I-D.draft-bre-openpgp-samples-01] and [I-D.draft-dkg-lamps-samples-02].

It may also include example renderings of these messages.

Author's Address

Daniel Kahn Gillmor
American Civil Liberties Union
125 Broad St.
New York, NY, 10004
United States of America

Email: dkg@fifthhorseman.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 13 May 2021

R. Housley
Vigil Security

9 November 2020

Using the AES-GMAC Algorithm with the Cryptographic Message Syntax (CMS)
draft-housley-lamps-cms-aes-mac-alg-00

Abstract

This document specifies the conventions for using the AES-GMAC Message Authentication Code algorithms with the Cryptographic Message Syntax (CMS) as specified in RFC 5652.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 13 May 2021.

Copyright Notice

Table of Contents

1. Introduction

This document specifies the conventions for using the AES-GMAC [AES][GCM] Message Authentication Code (MAC) algorithm with the Cryptographic Message Syntax (CMS) [RFC5652].

2. 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.

3. Message Authentication Code Algorithms

This section specifies the conventions employed by CMS [RFC5652] implementations that support the AES-GMAC [AES][GCM] Message Authentication Code (MAC) algorithm.

MAC algorithm identifiers are located in the AuthenticatedData macAlgorithm field.

MAC values are located in the AuthenticatedData mac field.

3.1. AES-GMAC

The AES-GMAC [AES][GCM] Message Authentication Code (MAC) algorithm uses one of the following algorithm identifiers; the choice depends on the size of the AES key, which is either 128 bits, 192 bits, or 256 bits:

```
aes OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16) us(840)
        organization(1) gov(101) csor(3) nistAlgorithm(4) 1 }

id-aes128-GMAC OBJECT IDENTIFIER ::= { aes 9 }

id-aes192-GMAC OBJECT IDENTIFIER ::= { aes 29 }

id-aes256-GMAC OBJECT IDENTIFIER ::= { aes 49 }
```

For all three of these algorithm identifier values, the AlgorithmIdentifier parameters field MUST be present, and the parameters MUST contain GMACParameters:

```
GMACParameters ::= SEQUENCE {
   nonce        OCTET STRING, -- recommended size is 12 octets
   length       MACLength DEFAULT 12 }

MACLength ::= INTEGER (12 | 13 | 14 | 15 | 16)
```

The GMACParameters nonce parameter is the GMAC initialization vector. The nonce may have any number of bits between 8 and 2^64, but it MUST be a multiple of 8 bits. Within the scope of any content-authentication key, the nonce value MUST be unique. A nonce value of 12 octets can be processed more efficiently, so that length for the nonce value is RECOMMENDED.

The GMACParameters length parameter field tells the size of the message authentication code. It MUST match the size in octets of the value in the AuthenticatedData mac field. A length of 12 octets is RECOMMENDED.

4. Implementation Considerations

An implementation of the Advanced Encryption Standard (AES) Galois/Counter Mode (GCM) authenticated encryption algorithm is specified in [GCM]. An implementation of AES-GCM can be used to compute the GMAC message authentication code by providing the content-authentication key as the AES key, the nonce as the initialization vector, a zero-length plaintext content, and the content to be authenticated as the additional authenticated data (AAD). The result of the AES-GCM invocation the AES-GMAC authentication code, which is called the authentication tag in some implementations. In AES-GCM, the encryption step is skipped when a zero-length input plaintext is provided; therefore, any value returned for ciphertext is ignored.

5. ASN.1 Module

The following ASN.1 module uses the definition for MAC-ALGORITHM from [RFC5912].

```
CryptographicMessageSyntaxGMACAlgorithms
    { iso(1) member-body(2) us(840) rsadsi(113549)
      pkcs(1) pkcs-9(9) smime(16) modules(0)
      id-mod-aes-gmac-alg-2020(TBD) }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

-- EXPORTS All

IMPORTS
  AlgorithmIdentifier{}, MAC-ALGORITHM
  FROM AlgorithmInformation-2009 -- from [RFC5912]
      { iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-algorithmInformation-02(58)} ;

-- Object Identifiers

aes OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16) us(840)
       organization(1) gov(101) csor(3) nistAlgorithm(4) 1 }

id-aes128-GMAC OBJECT IDENTIFIER ::= { aes 9 }

id-aes192-GMAC OBJECT IDENTIFIER ::= { aes 29 }

id-aes256-GMAC OBJECT IDENTIFIER ::= { aes 49 }

-- GMAC Parameters

GMACParameters ::= SEQUENCE {
   nonce        OCTET STRING, -- recommended size is 12 octets
   length       MACLength DEFAULT 12 }

MACLength ::= INTEGER (12 | 13 | 14 | 15 | 16)

-- Algorithm Identifiers

maca-aes128-GMAC MAC-ALGORITHM ::= {
   IDENTIFIER id-aes128-GMAC
   PARAMS TYPE GMACParameters ARE required
   IS-KEYED-MAC TRUE }
```

```
maca-aes192-GMAC MAC-ALGORITHM ::= {
   IDENTIFIER id-aes192-GMAC
   PARAMS TYPE GMACParameters ARE required
   IS-KEYED-MAC TRUE }

maca-aes256-GMAC MAC-ALGORITHM ::= {
   IDENTIFIER id-aes256-GMAC
   PARAMS TYPE GMACParameters ARE required
   IS-KEYED-MAC TRUE }

END -- of CryptographicMessageSyntaxGMACAlgorithms
```

6. IANA Considerations

IANA is asked to register object identifiers for one module identifier in the "SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)" registry for id-mod-aes-gmac-alg-2020.

7. Security Considerations

The CMS provides a method for authenticating data. This document identifies the conventions for using the AES-GMAC algorithm with the CMS.

The key management technique employed to distribute message-authentication keys must itself provide authentication, otherwise the content is delivered with integrity from an unknown source.

When more than two parties share the same message-authentication key, data origin authentication is not provided. Any party that knows the message-authentication key can compute a valid MAC, therefore the content could originate from any one of the parties.

Implementations must randomly generate message-authentication keys. The use of inadequate pseudo-random number generators (PRNGs) to generate keys can result in little or no security. An attacker may find it much easier to reproduce the PRNG environment that produced the keys, searching the resulting small set of possibilities, rather than brute force searching the whole key space. The generation of quality random numbers is difficult. [RFC4086] offers important guidance in this area.

Implementers should be aware that cryptographic algorithms become weaker with time. As new cryptanalysis techniques are developed and computing performance improves, the work factor to break a particular cryptographic algorithm will reduce. Therefore, cryptographic algorithm implementations should be modular allowing new algorithms to be readily inserted. That is, implementers should be prepared to regularly update the set of algorithms in their implementations.

8. References

8.1. Normative References

[AES] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)", FIPS Publication 197, November 2001.

[GCM] M., D., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST Special Publication 800-38D, November 2007.

[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>.

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

[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>.

8.2. Informative References

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[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>.

Author's Address

Russ Housley
Vigil Security, LLC
516 Dranesville Road
Herndon, VA, 20170
United States of America

Email: housley@vigilsec.com

Network Working Group
Internet-Draft
Updates: 4211 (if approved)
Intended status: Standards Track
Expires: 4 May 2021

R. Housley
Vigil Security

31 October 2020

Algorithm Requirements Update to the Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF) draft-housley-lamps-crmf-update-algs-01

Abstract

This document updates the cryptographic algorithm requirements for the Password-Based Message Authentication Code in the Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF) specified in RFC 4211.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 4 May 2021.

Table of Contents

1. Introduction

This document updates the cryptographic algorithm requirements for the Password-Based Message Authentication Code (MAC) in the Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF) [RFC4211]. The algorithms specified in [RFC4211] were appropriate in 2005; however, these algorithms are no longer considered the best choices. This update specifies algorithms that are more appropriate today.

2. 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.

3. Password-Based Message Authentication Code

Section 4.4 of [RFC4211] specifies a Password-Based MAC that relies on a one-way function to compute a symmetric key from the password and a MAC algorithm. This section specifies algorithm requirements for the one-way function and the MAC algorithm.

3.1. One-Way Function

Change the paragraph describing the "owf" as follows:

OLD:

> owf identifies the algorithm and associated parameters used to compute the key used in the MAC process. All implementations MUST support SHA-1.

NEW:

owf identifies the algorithm and associated parameters used to compute the key used in the MAC process. All implementations MUST support SHA-256 [SHS].

3.2. MAC Algorithm

Change the paragraph describing the "mac" as follows:

OLD:

mac identifies the algorithm and associated parameters of the MAC function to be used. All implementations MUST support HMAC-SHA1 [HMAC]. All implementations SHOULD support DES-MAC and Triple-DES-MAC [PKCS11].

NEW:

mac identifies the algorithm and associated parameters of the MAC function to be used. All implementations MUST support HMAC-SHA256 [HMAC]. All implementations SHOULD support AES-GMAC [GMAC] with a 128 bit key.

{{{ Note: Has an OID already been assigned for AES-GMAC? If not, we will need to do that too. }}}

4. IANA Considerations

This document makes no requests of the IANA.

5. Security Considerations

Cryptographic algorithms age; they become weaker with time. As new cryptanalysis techniques are developed and computing capabilities improve, the work required to break a particular cryptographic algorithm will reduce, making an attack on the algorithm more feasible for more attackers. While it is unknown how cryptoanalytic attacks will evolve, it is certain that they will get better. It is unknown how much better they will become or when the advances will happen. For this reason, the algorithm requirements for CRMF are updated by this specification.

When a Password-Based MAC is used, implementations must protect the password and the MAC key. Compromise of either the password or the MAC key may result in the ability of an attacker to undermine authentication.

6. Normative References

[AES] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)", FIPS Publication 197, November 2001.

[GMAC] M., D., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST Special Publication 800-38D, November 2007.

[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>.

[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>.

[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>.

[SHS] National Institute of Standards and Technology (NIST), "Secure Hash Standard", FIPS Publication 180-4, August 2015.

Author's Address

Russ Housley
Vigil Security, LLC
516 Dranesville Road
Herndon, VA, 20170
United States of America

Email: housley@vigilsec.com

LAMPS Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

H. Brockhaus
Siemens

November 2, 2020

CMP Algorithms
draft-ietf-lamps-cmp-algorithms-01

Abstract

This document describes the conventions for using several cryptographic algorithms with the Certificate Management Protocol (CMP). CMP is used to enroll and further manage the lifecycle of X.509 certificates.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 6, 2021.

Table of Contents

1. Introduction

[RFC Editor: please delete]: !!! The change history was moved to Appendix B !!!

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119]

[RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Message Digest Algorithms

This section specifies the conventions employed by CMP implementations that support SHA-1 or SHA2 algorithm family.

Digest algorithm identifiers are located in the hashAlg field of OOBCertHash, the owf field of Challenge, PBMParameter, and DHBMParameter, and the digestAlgorithms field of SignedData and the digestAlgorithm field of SignerInfo.

Digest values are located in the hashVal field of OOBCertHash, the witness field of Challenge, and the certHash field of CertStatus. In addition, digest values are input to signature algorithms.

2.1. SHA2

The SHA2 message digest algorithm family is defined in FIPS Pub 180-4 [FIPS180-4].

The message digest algorithms SHA-224, SHA-256, SHA-384, and SHA-512 produce a 224-bit are identified by the following object identifiers (OIDs):

```
id-sha224 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
   us(840) organization(1) gov(101) csor(3) nistalgorithm(4)
   hashalgs(2) 4 }
id-sha256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
   us(840) organization(1) gov(101) csor(3) nistalgorithm(4)
   hashalgs(2) 1 }
id-sha384 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
   us(840) organization(1) gov(101) csor(3) nistalgorithm(4)
   hashalgs(2) 2 }
id-sha512 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
   us(840) organization(1) gov(101) csor(3) nistalgorithm(4)
   hashalgs(2) 3 }
```

Further conventions to be considered are specified in RFC 5754 Section 2 [RFC5754].

3. Signature Algorithms

This section specifies the conventions employed by CMP implementations that support DSA, RSA, or ECDSA.

The signature algorithm is referred to as MSG_SIG_ALG in RFC 4210 Appendix D and E [RFC4210] and in the Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile].

Signature algorithm identifiers are located in the protectionAlg field of PKIHeader, the algorithmIdentifier field of POPOSigningKey, signatureAlgorithm field of p10cr, SignKeyPairTypes, and the SignerInfo signatureAlgorithm field of SignedData.

Signature values are located in the protection field of PKIMessage, signature field of POPOSigningKey, signature field of p10cr, and SignerInfo signature field of SignedData.

3.1. DSA

The DSA signature algorithm is defined in FIPS Pub 186-4 [FIPS186-4] and MAY be used with SHA-224 and SHA-256 as specified in RFC 5754 [RFC5754].

The algorithm identifiers for DSA with SHA2 signature values are:

```
id-dsa-with-sha224 OBJECT IDENTIFIER ::=  { joint-iso-ccitt(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   algorithms(4) id-dsa-with-sha2(3) 1 }
id-dsa-with-sha256 OBJECT IDENTIFIER ::=  { joint-iso-ccitt(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   algorithms(4) id-dsa-with-sha2(3) 2 }
```

Further conventions to be considered are specified in RFC 5754 Section 3.1 [RFC5754].

3.2. RSA

The RSA (RSASSA-PSS and RSASSA-PKCS1-v1_5) signature algorithm is defined in RFC 8017 [RFC8017]. RSASSA-PKCS1-v1_5 MAY be used with SHA-224, SHA-256, SHA-384, or SHA-512 as specified in RFC 5754 [RFC5754].

The algorithm identifiers for RSASAA-PSS signatures as specified in RFC 4055 [RFC4055] is:

```
id-RSASSA-PSS OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 10 }
```

Further conventions to be considered are specified in RFC 4056 [RFC4056].

The algorithm identifiers for RSASSA-PKCS1-v1_5 signatures as specified in RFC 4055 [RFC4055] are:

```
sha224WithRSAEncryption OBJECT IDENTIFIER ::= { iso(1)
   member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 14 }
sha256WithRSAEncryption  OBJECT IDENTIFIER  ::=  { iso(1)
   member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 11 }
sha384WithRSAEncryption  OBJECT IDENTIFIER  ::=  { iso(1)
   member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 12 }
sha512WithRSAEncryption  OBJECT IDENTIFIER  ::=  { iso(1)
   member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 13 }
```

Further conventions to be considered are specified in RFC 5754 Section 3.2 [RFC5754].

3.3. ECDSA

The ECDSA signature algorithm is defined in FIPS Pub 186-4 [FIPS186-4] and MAY be used with SHA-224, SHA-256, SHA-384, or SHA-512 as specified in RFC 5754 [RFC5754].

The algorithm identifiers for ECDSA with SHA2 signature values are:

```
ecdsa-with-SHA224 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) 1 }
ecdsa-with-SHA256 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) 2 }
ecdsa-with-SHA384 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) 3 }
ecdsa-with-SHA512 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) 4 }
```

Further conventions to be considered are specified in RFC 5754 Section 3.3 [RFC5754].

4. Key Management Algorithms

CMP accommodates the following general key management techniques: key agreement, key transport, and passwords.

CRMF [RFC4211] and CMP Updates [I-D.ietf-lamps-cmp-updates] facilitate the use of CMS [RFC5652] EnvelopedData by deprecating the use of EncryptedValue.

4.1. Key Agreement Algorithms

The key agreement algorithm is referred to as PROT_ENC_ALG in RFC 4210 Appendix D and E [RFC4210] and in the Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile].

Key agreement algorithms are only used in CMP when using CMS [RFC5652] EnvelopedData together with the key agreement key management technique. When a key agreement algorithm is used, a key-encryption algorithm (Section 4.3) is needed next to the content-encryption algorithm (Section 5).

Key agreement algorithm identifiers are located in the EnvelopedData RecipientInfos KeyAgreeRecipientInfo keyEncryptionAlgorithm fields.

Key encryption algorithm identifiers are located in the EnvelopedData RecipientInfos KeyAgreeRecipientInfo keyEncryptionAlgorithm field.

Wrapped content-encryption keys are located in the EnvelopedData RecipientInfos KeyAgreeRecipientInfo RecipientEncryptedKeys encryptedKey field.

4.1.1. Diffie-Hellman

Diffie-Hellman key agreement is defined in RFC 2631 [RFC2631] and MAY be used in the ephemeral-static or a static-static variant as specified in RFC 3370 [RFC3370].

The Diffie-Hellman algorithm identifiers are:

```
id-alg-ESDH OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3) 5 }
id-alg-SSDH OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3) 10 }
```

Further conventions to be considered are specified in RFC 3370 Section 4.1 [RFC3370].

4.1.2. ECDH

Elliptic Curve Diffie-Hellman (ECDH) key agreement is defined in RFC 5753 [RFC5753] and MAY be used on the ephemeral-static variant in RFC 5753 [RFC5753], the 1-Pass ECMQV variant as specified in RFC 5753 [RFC5753] or the static-static variant as specified in RFC RFC 6278 [RFC6278].

Algorithm Identifiers and further conventions to be considered are specified in RFC RFC 5753 [RFC5753] and RFC 6278 [RFC6278].

4.2. Key Transport Algorithms

The key transport algorithm is also referred to as PROT_ENC_ALG in RFC 4210 Appendix D and E [RFC4210] and in the Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile].

Key transport algorithms are only used in CMP when using CMS [RFC5652] EnvelopedData together with the key transport key management technique.

Key transport algorithm identifiers are located in the EnvelopedData RecipientInfos KeyTransRecipientInfo keyEncryptionAlgorithm field.

Key transport encrypted content-encryption keys are located in the EnvelopedData RecipientInfos KeyTransRecipientInfo encryptedKey field.

4.2.1. RSA

The RSA key transport algorithm is the RSA encryption scheme defined in RFC 8017 [RFC8017].

The algorithm identifier for RSA (PKCS #1 v1.5) is:

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

The algorithm identifier for RSAES-OAEP is:

```
id-RSAES-OAEP  OBJECT IDENTIFIER  ::=  { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 7 }
```

Further conventions to be considered for PKCS #1 v1.5 are specified in RFC 3370 Section 4.2.1 [RFC3370] and for RSAES-OAEP in RFC 3560 [RFC3560].

4.3. Symmetric Key-Encryption Algorithms

The symmetric key-encryption algorithm is also referred to as PROT_SYM_ALG in RFC 4210 Appendix D and E [RFC4210] and in the Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile].

As symmetric key-encryption key management technique is not used by CMP, the symmetric key-encryption algorithm is only needed when using the key agreement or password-based key management technique with CMS [RFC5652] EnvelopedData.

Key-encryption algorithm identifiers are located in the EnvelopedData RecipientInfos KeyAgreeRecipientInfo keyEncryptionAlgorithm and EnvelopedData RecipientInfos PassworRecipientInfo keyEncryptionAlgorithm fields.

Wrapped content-encryption keys are located in the EnvelopedData RecipientInfos KeyAgreeRecipientInfo RecipientEncryptedKeys encryptedKey and EnvelopedData RecipientInfos PassworRecipientInfo encryptedKey fields.

4.3.1. AES Key Wrap

The AES encryption algorithm is defined in FIBS Pub 197 [FIPS197] and the key wrapping is defined in RFC 3394 [RFC3394].

AES key encryption has the algorithm identifier:

```
id-aes128-wrap OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) aes(1) 5 }
id-aes192-wrap OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) aes(1) 25 }
id-aes256-wrap OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) aes(1) 45 }
```

Further conventions to be considered for AES key wrap are specified in RFC 3394 Section 2.2 [RFC3394] and RFC 3565 Section 2.3.2 [RFC3565].

4.4. Key Derivation Algorithms

Key derivation algorithms are only used in CMP when using CMS [RFC5652] EnvelopedData together with password-based key management technique.

Key derivation algorithm identifiers are located in the EnvelopedData RecipientInfos PassworRecipientInfo keyDerivationAlgorithm field.

4.4.1. Password-based Key Derivation Function 2

The password-based key derivation function 2 (PBKDF2) is defined in RFC 8018 [RFC8018].

Password-based key derivation function 2 has the algorithm identifier:

```
id-PBKDF2 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
   rsadsi(113549) pkcs(1) pkcs-5(5) 12 }
```

Further conventions to be considered for PBKDF2 are specified in RFC 3370 Section 4.4.1 [RFC3370] and RFC 8018 Section 5.2 [RFC8018].

5. Content Encryption Algorithms

The content encryption algorithm is also referred to as PROT_SYM_ALG in RFC 4210 Appendix D and E [RFC4210] and in the Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile].

Content encryption algorithms are only used in CMP when using CMS [RFC5652] EnvelopedData to transport a signed private key package in case of central key generation or key archiving, a certificate to facilitate implicit prove-of-possession, or a revocation passphrase in encrypted form.

Content encryption algorithm identifiers are located in the EnvelopedData EncryptedContentInfo contentEncryptionAlgorithmrithm field.

Encrypted content is located in the EnvelopedData EncryptedContentInfo encryptedContent field.

5.1. AES

The AES encryption algorithm is defined in FIPS Pub 197 [FIPS197]. Details of usage of AES-CCM and AES-GCM in CMS [RFC5652] EnvelopedData is specified in RFC 5084 [RFC5084].

AES content encryption has the algorithm identifier:

```
id-aes128-CCM OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) aes(1) 7 }
id-aes192-CCM OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) aes(1) 27 }
id-aes256-CCM OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) aes(1) 47 }
id-aes128-GCM OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) aes(1) 6 }
id-aes192-GCM OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) aes(1) 26 }
id-aes256-GCM OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) aes(1) 46 }
```

Further conventions to be considered for AES content encryption are specified in RFC 5084 [RFC5084].

6. Message Authentication Code Algorithms

The message authentication code algorithm is also referred to as MSG_MAC_ALG in RFC 4210 Appendix D and E [RFC4210] and in the Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile].

Message authentication code algorithm identifiers are located in the mac field of PBMParameter and DHBMParameter, the PBKDF2-params prf field.

Message authentication code values are located in the EnvelopedData EncryptedContentInfo encryptedContent field.

6.1. Password-based MAC

The password-based MAC is defined in RFC 4210 [RFC4210].

The algorithm identifier for password-based MAC as specified in RFC 4210 [RFC4210] is:

```
id-PasswordBasedMac OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) nt(113533) nsn(7) algorithms(66) 13 }
```

Further conventions to be considered for password-based MAC are specified in RFC 4210 Section 5.1.3.1 [RFC4210].

6.2. Diffie-Hellman-based MAC

The Diffie-Hellman-based MAC is defined in RFC 4210 [RFC4210].

The algorithm identifiers for Diffie-Hellman-based MAC is:

```
id-DHBasedMac OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) nt(113533) nsn(7) algorithms(66) 30 }
```

Further conventions to be considered for Diffie-Hellman-based MAC are specified in RFC 4210 Section 5.1.3.2 [RFC4210].

6.3. SHA2-based HMAC

The HMAC is defined in RFC 2104 [RFC2104] and FIPS Pub 198-1 [FIPS198-1]. The SHA2 algorithms are defined in Section 2.1Section 2.1 and FIPS Pub 180-4 [FIPS180-4].

The algorithm identifiers for SHA2-based HMAC as specified in RFC 4231 [RFC4231] are:

```
id-hmacWithSHA224 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) digestAlgorithm(2) 8 }
id-hmacWithSHA256 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) digestAlgorithm(2) 9 }
id-hmacWithSHA384 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) digestAlgorithm(2) 10 }
id-hmacWithSHA512 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) digestAlgorithm(2) 11 }
```

Further conventions to be considered for SHA2-based HMAC are specified in RFC 4231 Section 3.1 [RFC4231].

7. IANA Considerations

This document does not request changes to the IANA registry.

8. Security Considerations

RFC 4210 Appendix D.2 [RFC4210] contains a set of algorithms, mandatory to be supported by conforming implementations. Theses algorithms were appropriate at the time CMP war releases, but as cryptographic algorithms weaken over time, some of them should not be uses anymore. In general, new attacks are emerging due to research

cryptoanalysis or increase in computing power. new algorithms were introduced that are more resistant to today's attacks.

This document lists many cryptographic algorithms usable with CMP to offer implementers a more up to date choice. Finally, the algorithms to be supported also heavily depend on the utilizes certificates in the target environment.

In the appendix of this document there is also an update to the Appendix D.2 of RFC 4210 [RFC4210] and a set of algorithms to be supported when implementing the Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile].

To keep the list of algorithms to be used with CMP up to date to enlist secure algorithms resisting known attack scenarios, future algorithms should be added and weakened algorithms should be deprecated.

9. Acknowledgements

Thanks to Russ Housley for his input and feedback to this document.

10. References

10.1. Normative References

[FIPS180-4]
NIST, "FIPS Pub 180-4: Secure Hash Standard (SHA)", August 2015 , <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

[FIPS186-4]
NIST, "FIPS Pub 186-4: Digital Signature Standard (DSS)", July 2013, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>.

[FIPS197] NIST, "FIPS Pub 197: Advanced Encryption Standard (AES)", November 2001, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>.

[FIPS198-1]
NIST, "The Keyed-Hash Message Authentication Code (HMAC)", July 2008, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.198-1.pdf>.

[I-D.ietf-lamps-cmp-updates]
Brockhaus, H., "CMP Updates", draft-ietf-lamps-cmp-updates-05 (work in progress), September 2020.

[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>.

[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>.

[RFC2631] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, DOI 10.17487/RFC2631, June 1999, <https://www.rfc-editor.org/info/rfc2631>.

[RFC3370] Housley, R., "Cryptographic Message Syntax (CMS) Algorithms", RFC 3370, DOI 10.17487/RFC3370, August 2002, <https://www.rfc-editor.org/info/rfc3370>.

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394, September 2002, <https://www.rfc-editor.org/info/rfc3394>.

[RFC3560] Housley, R., "Use of the RSAES-OAEP Key Transport Algorithm in Cryptographic Message Syntax (CMS)", RFC 3560, DOI 10.17487/RFC3560, July 2003, <https://www.rfc-editor.org/info/rfc3560>.

[RFC3565] Schaad, J., "Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)", RFC 3565, DOI 10.17487/RFC3565, July 2003, <https://www.rfc-editor.org/info/rfc3565>.

[RFC4055] Schaad, J., Kaliski, B., and R. Housley, "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 4055, DOI 10.17487/RFC4055, June 2005, <https://www.rfc-editor.org/info/rfc4055>.

[RFC4056] Schaad, J., "Use of the RSASSA-PSS Signature Algorithm in Cryptographic Message Syntax (CMS)", RFC 4056, DOI 10.17487/RFC4056, June 2005, <https://www.rfc-editor.org/info/rfc4056>.

[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>.

[RFC4231] Nystrom, M., "Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512", RFC 4231, DOI 10.17487/RFC4231, December 2005, <https://www.rfc-editor.org/info/rfc4231>.

[RFC5084] Housley, R., "Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS)", RFC 5084, DOI 10.17487/RFC5084, November 2007, <https://www.rfc-editor.org/info/rfc5084>.

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

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

[RFC5754] Turner, S., "Using SHA2 Algorithms with Cryptographic Message Syntax", RFC 5754, DOI 10.17487/RFC5754, January 2010, <https://www.rfc-editor.org/info/rfc5754>.

[RFC6278] Herzog, J. and R. Khazan, "Use of Static-Static Elliptic Curve Diffie-Hellman Key Agreement in Cryptographic Message Syntax", RFC 6278, DOI 10.17487/RFC6278, June 2011, <https://www.rfc-editor.org/info/rfc6278>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[RFC8018] Moriarty, K., Ed., Kaliski, B., and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1", RFC 8018, DOI 10.17487/RFC8018, January 2017, <https://www.rfc-editor.org/info/rfc8018>.

[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>.

10.2. Informative References

[I-D.ietf-lamps-lightweight-cmp-profile] Brockhaus, H., Fries, S., and D. Oheimb, "Lightweight CMP Profile", draft-ietf-lamps-lightweight-cmp-profile-03 (work in progress), October 2020.

Appendix A. Algorithm Use Profiles

This appendix provides profiles of algorithms and respective conventions for different application use cases.

A.1. Algorithm Profile for PKI Management Message Profiles

The following table contains definitions of algorithm used within PKI Management Message Profiles as defined in CMP Appendix D.2 [RFC4210]. The columns in the table are:

Name: an identifier used for message profiles

Use: description of where and for what the algorithm is used

Mandatory: an AlgorithmIdentifier which MUST be supported by conforming implementations

Name	Use	Mandatory
MSG_SIG_ALG	protection of PKI messages using signature	RSA
MSG_MAC_ALG	protection of PKI messages using MACing	PasswordBasedMac
SYM_PENC_ALG	symmetric encryption of an end entity's private key where symmetric key is distributed out-of-band	AES-wrap
PROT_ENC_ALG	asymmetric algorithm used for encryption of (symmetric keys for encryption of) private keys transported in PKIMessages	D-H
PROT_SYM_ALG	symmetric encryption algorithm used for encryption of private key bits (a key of this type is encrypted using PROT_ENC_ALG)	AES

Mandatory Algorithm Identifiers and Specifications:

RSA: sha256WithRSAEncryption with 2048 bit, see Section 3.2

PasswordBasedMac: id-PasswordBasedMac, see Section 6.1 (with id-sha256 as the owf parameter, see Section 2.1 and id-hmacWithSHA256 as the mac parameter, see Section 6.3)

D-H: id-alg-ESDH, see Section 4.1.1

AES-wrap: id-aes256-wrap, see Section 4.3.1

AES: id-aes256-GCM, see Section 5.1

A.2. Algorithm Profile for Lightweight CMP Profile

The following table contains definitions of algorithm which MUST be supported by conforming implementations This profile is referenced in the Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile]. The columns in the table are:

Name: an identifier used for message profiles

Use: description of where and for what the algorithm is used

Mandatory: an AlgorithmIdentifier which MUST be supported by conforming implementations

Name	Use	Mandatory
MSG_SIG_ALG	protection of PKI messages using signature	ECDSA
MSG_MAC_ALG	protection of PKI messages using MACing	PasswordBasedMac
KM_KA_ALG	asymmetric key agreement algorithm used for agreement of a symmetric keys for encryption of EnvelopedData, e.g., a private key transported in PKIMessages	ECDH
KM_KT_ALG	asymmetric key encryption algorithm used for transport of a symmetric keys for encryption of EnvelopedData, e.g., a private key transported in PKIMessages	RSA
KM_PB_ALG	symmetric derivation algorithm used to derive a symmetric key for encryption of EnvelopedData, e.g., a private key transported in PKIMessages, from a password	PBKDF2
PROT_ENC_ALG	Symmetric key encryption algorithm to encrypt a content encryption key	AES-wrap
PROT_SYM_ALG	symmetric content encryption algorithm used for encryption of, e.g., private key bits (a key of this type is encrypted using PROT_ENC_ALG)	AES

Mandatory Algorithm Identifiers and Specifications:

< TBD: The list of mandatory algorithms has to be defined later. >

Appendix B. History of changes

Note: This appendix will be deleted in the final version of the document.

From version 00 -> 01:

- Changed sections Symmetric Key-Encryption Algorithms and Content Encryption Algorithms based on the discussion on the mailing list (see thread "[CMP Algorithms] Use Key-Wrap with or without padding in Section 4.3 and Section 5")

- Added Appendix A with updated algorithms profile for RDC4210 Appendix D.2 and first proposal for the Lightweight CMP Profile

- Minor changes in wording

Author's Address

Hendrik Brockhaus
Siemens AG

Email: hendrik.brockhaus@siemens.com

LAMPS Working Group
Internet-Draft
Updates: 4210, 6712 (if approved)
Intended status: Standards Track
Expires: May 6, 2021

H. Brockhaus
Siemens

November 2, 2020

CMP Updates
draft-ietf-lamps-cmp-updates-06

Abstract

This document contains a set of updates to the base syntax and transport of Certificate Management Protocol (CMP) version 2. This document updates RFC 4210 and RFC 6712.

Specifically, the CMP services updated in this document comprise the enabling of using EnvelopedData instead of EncryptedValue, adding new general message types, the definition of extended key usages to identify certificates of CMP endpoints on certification and registration authorities, and adds an HTTP URI discovery mechanism and extend the URI structure.

Table of Contents

1. Introduction

[RFC Editor: please delete]: !!! The change history was moved to Appendix B !!!

While using CMP [RFC4210] in industrial and IoT environments and developing the Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile] some limitations were identified in the original CMP specification. This document updates RFC 4210 [RFC4210] and RFC 6712 [RFC6712] to overcome these limitations.

In general, this document aims to improve the crypto agility of CMP to be flexible to react on future advances in cryptography.

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

1.1. Convention and Terminology

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

Technical terminology is used in conformance with RFC 4210 [RFC4210], RFC 4211 [RFC4211], and RFC 5280 [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 co-located 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 RFC 4210 [RFC4210]. They are always related to the base specification. Hence references to the original sections in RFC 4210 [RFC4210] are used whenever possible.

Insert this section at the end of the current Section 1.

1.1 Changes since RFC 4210

The following updates are made in [thisRFC]:

- Add new extended key usages for different 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 to act as the indicated PKI management entity.

- Extend 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 extend crypto agility in CMP. Note that according to RFC 4211 [RFC4211] section 2.1.9 the use of the EncryptedValue structure has been deprecated in favor of the EnvelopedData structure. RFC 4211 [RFC4211] offers the EncryptedKey structure, a choice of EncryptedValue and EnvelopedData for migration to EnvelopedData. For reasons of completeness and consistency the exchange of EncryptedValue is performed for all usages in RFC 4210 [RFC4210]. This includes the protection of centrally generated private keys, encryption of certificates, and revocation passphrases.

- Adding new general message types to request CA certificates, a root CA update, or a certificate request template.

- Extend the usage of polling also to p10cr messages.

< TBD: The specification of algorithm profiles seed to be moved to a separate document. >

2.2. New Section 4.5 - Extended Key Usage

The following subsection describes new extended key usages for different CMP server types specified in RFC 4210 [RFC4210].

Insert this section at the end of the current Section 4.

4.5 Extended Key Usage

The Extended Key Usage (EKU) extension indicates the purposes for which the certified public key may be used. It therefore restricts the use of a certificate to specific applications.

A CA may want to delegate parts of their duties to other PKI management entities. The mechanism to prove this delegation explained in this section offers zero-touch means to check the authorization of such delegation. Such delegation could also be expressed by other means, e.g., explicit configuration.

To offer automatic validation means for the delegation of a role by a CA, the certificates used by PKI management entities 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 the delegating CA to act as the PKI management entity as described below.

The ASN.1 to define these EKUs is:

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

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

Note: RFC 6402 section 2.10 [RFC6402] specifies OIDs for a CMC CA and a CMC RA. As the functionality of a CA and RA is not specific to whether use CMC or CMP as certificate management protocol, the same OIDs SHALL be used for a CMP CA and a CMP RA.

< TBD: The Description of the OIDs for id-kp-cmcCA and id-kp-cmcRA needs to be extended to avoid confusion as they currently only refer to CMC. >

The description of the PKI management entity for each of the EKUs is as follows:

CMP CA: CMP Certification Authorities are CMP endpoints on CA equipment as described in section 3.1.1.2. The key used in the context of CMP management operations, especially CMP message protection, need not be the same key that signs the certificates. It is necessary, however, to ensure that the entity acting as CMP CA is authorized to do so. Therefore, the CMP CA MUST do one of the following,

* use the CA private key on the CMP endpoint, or

* explicitly designate this authority to another entity.

For automatic validation of such delegation it MUST be indicated by the id-kp-cmcCA extended key usage. This extended key usage MUST be placed into the certificate used on the CA equipment and the CA that delegates this role MUST issue the CMP CA certificate.

Note: Using a separate key pair for protecting CMP management operations at the CA decreases the number of operations of the private key used to sign certificates.

CMP RA: CMP Registration Authorities are CMP endpoints on RA equipment as described in Section 3.1.1.3. A CMP RA is identified by the id-kp-cmcRA extended key usage. This extended key usage is placed into RA certificates. The CA that delegated this role is identified by the CA that issued the CMP RA certificate.

CMP KGA: CMP Key Generation Authorities are identified by the id-kp-cmKGA extended key usage. Though 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. This authorization is either with the CA certificate itself, or indicated by placing the id-kp-cmKGA extended key usage into the CMP RA or CMP CA certificate used to authenticate the origin of the private key, and to express the authorization to offer this service.

Note: In device PKIs, especially those issuing IDevID certificates, CA may have very long validity (including the GeneralizedTime value 99991231235959Z to indicate a not well-defined expiration date as specified in IEEE 802.1AR Section 8.5 [IEEE802.1AR] and RFC 5280 Section 4.1.2.5 [RFC5280]). Such validity periods SHOULD NOT be used for protection of CMP messages. Certificates for delegated CMP message protection (CMP CA, CMP RA, CMP KGA) MUST NOT use indefinite expiration date.

2.3. Replace Section 5.1.3.4 - Multiple Protection

Section 5.1.3.4 of RFC 4210 [RFC4210] describes the nested message. This document opens the usage of nested messages also for batch transport of PKI messages between different PKI management entities.

Replace the text of the section with the following text.

In cases where an end entity sends a protected PKI message to an RA, the RA MAY forward that message to a CA, adding its own protection (which MAY be a MAC or a signature, depending on the information and certificates shared between the RA and the CA). There are different use cases for such multi protected messages.

- The RA confirms the validation and authorization of a message and forwards the original message unchanged.

- The RA collects several messages and forwards them in a batch. This can for instance be used to bridge an off-line connection between two PKI management entities. In communication to the CA request messages and in communication from the CA response or announcement messages will be collected in such batch.

- The RA modifies the message(s) in some way (e.g., add or modify particular field values or add new extensions) before forwarding them, then it MAY create its own desired PKIBody. In case the changes made by the RA to PKIMessage breaks the POP, the RA MUST either set the POP RAVerified or include the original PKIMessage from the EE in the generalInfo field of PKIHeader of the nested message (to force the CA to check POP on the original 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 requests in PKIBody). For simplicity reasons, if batching is used in combination with inclusion of the original PKIMessage in the generalInfo field, all messages in the batch MUST be of the same type (e.g., ir).

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

```
NestedMessageContent ::= PKIMessages
```

(The use of PKIMessages, a SEQUENCE OF PKIMessage, lets the RA batch the requests of several EEs in a single new message.)

2.4. Replace Section 5.2.2. - Encrypted Values

Section 5.2.2 of RFC 4210 [RFC4210] describes the usage 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.

Where encrypted data (restricted, in this specification, to be either private keys, certificates, or passwords) are sent in PKI messages, the EncryptedKey data structure is used.

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

See CRMF [RFC4211] for EncryptedKey and EncryptedValue syntax and for EnvelopedData syntax see CMS [RFC5652]. Using the EncryptedKey data structure, the choice to either use EncryptedValue (for backward compatibility only) or EnvelopedData is offered. The use of the EncryptedValue structure has been deprecated in favor of the EnvelopedData structure. Therefore, it is recommended to use EnvelopedData.

Note: As we reuse the EncryptedKey structure defined in CRMF [RFC4211], the update is backward compatible. Using the new syntax with the untagged default choice EncryptedValue is bitwise compatible with the old syntax.

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

EnvelopedData is used as follows:

- Contains only one recepientInfo structure because the content is encrypted only for one recipient.

- Contains a private key in the AsymmetricKeyPackage structure as defined in RFC 5958 [RFC5958] wrapped in a SignedData structure as specified in CMS section 5 [RFC5652] signed by the Key Generation Authority.

- Contains a certificate or revocation passphrase directly in the encryptedContent field.

Note: To ensure explicit control of the encoding of the private key according to the specific algorithm the new key pair in an asymmetric key package structure as specified in [RFC5958].

The content of the EnvelopedData structure, as specified in CMS section 6 [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 for the recipient:

- Recipient's certificate that contains a key usage extension asserting keyAgreement: The content-encryption key will be protected using the key agreement key management technique, as specified in CMS section 6.2.2 [RFC5652].

- Recipient's certificate that contains a key usage extension asserting keyEncipherment: The content-encryption key will be protected using the key transport key management technique, as specified in CMS section 6.2.1 [RFC5652].

- Jointly shared secret: The content-encryption key will be protected using the password-based key management technique, as specified in CMS section 6.2.4 [RFC5652].

2.5. Update Section 5.3.4. - Certification Response

Section 5.3.4 of RFC 4210 [RFC4210] describes the Certification Response. This document updates the syntax by using the parent structure EncryptedKey instead of EncryptedValue as described in Section 2.1 above.

Replace the ASN.1 syntax of CertifiedKeyPair and CertOrEncCert with the following text.

```
CertifiedKeyPair ::= SEQUENCE {
    certOrEncCert       CertOrEncCert,
    privateKey      [0] EncryptedKey        OPTIONAL,
    -- see [CRMF] for comment on encoding
    publicationInfo [1] PKIPublicationInfo  OPTIONAL
}

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

Add the following paragraphs to the end of the section.

The use of EncryptedKey is described in section 5.2.2.

2.6. Update Section 5.3.19.2. - Signing Key Pair Types

The following section clarifies the usage of the Signing Key Pair Types PKI general message on referencing EC curves.

Insert this note at the end of Section 5.3.19.2.

Note: In case you wish to offer several EC curves, you need to put several id-ecPublicKey elements, one each per named curve.

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

The following section clarifies the usage of the Encryption/Key Agreement Key Pair Types PKI general message on referencing EC curves.

Insert this note at the end of Section 5.3.19.3.

Note: In case you wish to offer several EC curves, you need to put several id-ecPublicKey elements, one each per named curve.

2.8. Replace Section 5.3.19.9. - Revocation Passphrase

Section 5.3.19.9 of RFC 4210 [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 2.1 above.

Replace the text of the section with the following text.

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.

2.9. New Section 5.3.19.14 - CA Certificates

The following subsection describes the PKI general messages using id-it-caCerts. The use is specified in in Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile] Section 4.4.

Insert this section after Section 5.3.19.13.

2.3.19.14 CA Certificates

This MAY be used by the client to get the latest CA intermediate and issuing CA certificates.

```
GenMsg:    {id-it 17}, < absent >
GenRep:    {id-it 17}, SEQUENCE OF CMPCertificate | < absent >
```

2.10. New Section 5.3.19.15 - Root CA Certificates Update

The following subsection describes the PKI general messages using id-it-rootCaKeyUpdate. The use is specified in in Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile] Section 4.4.

Insert this section after new Section 5.3.19.14.

5.3.19.15. Root CA Certificates Update

This MAY be used by the client to get an update of an existing root CA Certificate.

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

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.

2.11. New Section 5.3.19.16 - Certificate Request Template

The following subsection describes the PKI general messages using id-it-certReqTemplate. The use is specified in in Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile] Section 4.4.

Insert this section after new Section 5.3.19.15.

5.3.19.16. Certificate Request Template

This MAY be used by the client to get a template with parameters for a future certificate request operation.

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

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

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

   id-regCtrl-algId OBJECT IDENTIFIER ::= { id-regCtrl TBD3 }
   AlgIdCtrl ::= AlgorithmIdentifier

   id-regCtrl-rsaKeyLen OBJECT IDENTIFIER ::= { id-regCtrl TBD4 }
   RsaKeyLenCtrl ::= Integer
```

< TBD: The OIDs TBD3 and TBD4 have to be registered at IANA. >

CertReqTemplateValue contains a prefilled certTemplate to be used for the future certificate request. The SubjectPublicKeyInfo field in the certTemplate MUST NOT be used. In case the PKI management entity wishes to specify supported algorithms, the controls field MUST be used. One AttributeTypeAndValue per supported algorithm MUST be used.

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

2.12. Update Section 5.3.22 - Polling Request and Response

Section 5.3.22 of RFC 4210 [RFC4210] describes when and how polling messages are used. This document adds the polling mechanism also to outstanding p10cr transactions.

Replace all paragraphs in front of the state machine diagram with the following text.

This pair of messages is intended to handle scenarios in which the client needs to poll the server in order to determine the status of an outstanding ir, cr, p10cr, or kur transaction (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 }
```

The following clauses describe when polling messages are used, and how they 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, following the ack, a pollReq for all pending certificates.

2 In response to a pollReq, a CA/RA will return an ip, cp, or kup if one or more of the pending certificates is ready; otherwise, it will return a pollRep.

3 If the EE receives a pollRep, it will wait for at least as long as the checkAfter value before sending another pollReq.

4 If an ip, cp, or kup is received in response to a pollReq, then it will be treated in the same way as the initial response.

Note: A p10cr message contains exactly one CertificationRequestInfo data structure as specified in PKCS#10 [RFC2986] but no certificate request number. Therefore, the certReqId MUST be set to 0 in all following messages of this transaction.

2.13. Update Section 9 - IANA Considerations

Section 9 of RFC 4210 [RFC4210] contains the IANA Considerations of that document. As this document defines a new and updates two existing Extended Key Usages, the IANA Considerations need to be updated accordingly.

Add the following paragraphs between the first and second paragraph of the section.

Within the SMI-numbers registry "SMI Security for PKIX Extended Key Purpose Identifiers (1.3.6.1.5.5.7.3)" (see https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-1.3.6.1.5.5.7.3) as defined in RFC 7299 [RFC7299] three changes have been performed.

Two existing entries have been updated to also point to this document:

Decimal	Description	References
27	id-kp-cmcCA	[RFC6402][thisRFC]
28	id-kp-cmcRA	[RFC6402][thisRFC]

One new entry has been added:

Decimal	Description	References
32	id-kp-cmKGA	[thisRFC]

Within the SMI-numbers registry "SMI Security for PKIX CMP Information Types (1.3.6.1.5.5.7.4)" (see https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-1.3.6.1.5.5.7.4) as defined in RFC 7299 [RFC7299] three changes have been performed.

Three new entry have been added:

Decimal	Description	References
17	id-it-caCerts	[thisRFC]
18	id-it-rootCaKeyUpdate	[thisRFC]
19	id-it-certReqTemplate	[thisRFC]

WWithin the SMI-numbers registry " SMI Security for PKIX CRMF Registration Controls (1.3.6.1.5.5.7.5.1)" (see https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-1.3.6.1.5.5.7.5.1) as defined in RFC 7299 [RFC7299] two changes have been performed.

Two new entry have been added:

Decimal	Description	References
TBD3	id-regCtrl-algId	[thisRFC]
TBD4	id-regCtrl-rsaKeyLen	[thisRFC]

2.14. Update Appendix B - The Use of Revocation Passphrase

Appendix B of RFC 4210 [RFC4210] describes the usage of the revocation passphrase. As this document updates RFC 4210 [RFC4210] to utilize the parent structure EncryptedKey instead of EncryptedValue as described in Section 2.1 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 of RFC 4210 [RFC4210] 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 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 (i.e., for use of EnvelopedData this is in the decrypted bytes of encryptedContent field and for use of EncryptedValue this is in the decrypted bytes of the encValue field) to the relevant CA/RA; furthermore, the transfer is accomplished with appropriate confidentiality characteristics.

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

- When using EnvelopedData the localKeyId attribute as specified in RFC 2985 [RFC2985] and when using EncryptedValue the valueHint field 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).

2.15. Update Appendix C - Request Message Behavioral Clarifications

Appendix C of RFC 4210 [RFC4210] provides clarifications to the request message behavior. As this document updates RFC 4210 [RFC4210] to utilize the parent structure EncryptedKey instead of EncryptedValue as described in Section 2.1 above, the description is updated accordingly.

Replace the note coming after the ASN.1 syntax of POPOPrivKey of this section with the following text.

```
-- **********
-- *  the type of "thisMessage" is given as BIT STRING in RFC 4211
-- *  [RFC4211]; it should be "EncryptedKey" (in accordance with
-- *  Section 5.2.2 of this specification). Therefore, this document
-- *  makes the behavioral clarification of specifying that the
-- *  contents of "thisMessage" MUST be encoded either as
-- *  "EnvelopedData" or "EncryptedValue" (only for backward
-- *  compatibility) and then wrapped in a BIT STRING.  This allows
-- *  the necessary conveyance and protection of the private key
-- *  while maintaining bits-on-the-wire compatibility with RFC 4211
-- *  [RFC4211].
-- **********
```

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

Appendix D.2 of RFC 4210 [RFC4210] provides a list of Algorithms implementations must support when claiming conformance with PKI Management Message Profiles as specified in CMP Appendix D.2 [RFC4210].

Replace the text of the section with the following text.

For specifications of algorithms identifiers and respective conventions for conforming implementations, please refer to CMP Algorithms Appendix A.1 [I-D.ietf-lamps-cmp-algorithms].

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

Appendix D.4 of RFC 4210 [RFC4210] provides the initial registration/ certification scheme. This scheme shall continue to use EncryptedValue for backward compatibility reasons.

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. New Section 1.1. - Changes since RFC 6712

The following subsection describes feature updates to RFC 6712 [RFC6712]. They are always related to the base specification. Hence references to the original sections in RFC 6712 [RFC6712] are used whenever possible.

Insert this section at the end of the current Section 1.

1.1 Changes since RFC 6712

The following updates are made in draft-ietf-lamps-cmp-updates:

- Add an HTTP URI discovery mechanism and extend the URI structure.

3.2. Replace Section 3.6. - HTTP Request-URI

Section 3.6 of RFC 6712 [RFC6712] specifies the used HTTP URIs. This document adds a discovery mechanism and extends the URIs.

Replace the text of the section with the following text.

Each CMP server on a PKI management entity supporting HTTP or HTTPS transport MUST support the use of the path-prefix of '/.well-known/' as defined in RFC 8515 [RFC8515] and the registered name of '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 operational path of the PKI management entity. Additional arbitrary label, e.g., 'profileLabel' and 'operationLabel', may be configured as a separate component or as part of the full operational path to provide further information. The 'profileLabel' may support addressing multiple CAs or certificate profiles and the 'operationLabel' may support addressing PKI management operation specific endpoints. A valid full operational path can look like this:

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

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

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

4 http://www.example.com/.well-known/cmp/profileLabel/operationLabel

The discovery of supported endpoints as defined above will provide the information to the CMP client how to contact the PKI management entity and, if available, how to request enrolment for a specific certificate profile or revoke a certificate at a specific CA.

Querying the PKI management entity, the CMP client will get a list of potential endpoints supported by the PKI management entity.

Performing a GET on "/.well-known/cmp" to the default port MUST return a set of links to endpoints available from the CMP server. In addition to the link also the expected format of the data object is provided as content type (ct).

< TBD: It needs to be discussed if the discovery should be performed using GET on "/.well-known/cmp" or GET on "/.well-known" only. >

The following provides an illustrative example for a PKI management entity supporting various PKI management operations for various certificate profiles and CAs.

Detailed message description:

REQ: GET /.well-known/cmp

RES: Content
</cmp/certprofile1/operation1>;ct=pkixcmp
</cmp/certprofile2/operation1>;ct=pkixcmp
</cmp/certprofile3/operation1>;ct=pkixcmp
</cmp/certprofile1/operation2>;ct=pkixcmp
</cmp/certprofile2/operation2>;ct=pkixcmp
</cmp/certprofile3/operation2>;ct=pkixcmp
</cmp/ca1/operation3>;ct=pkixcmp
</cmp/ca2/operation3>;ct=pkixcmp

3.3. Update Section 6. - IANA Considerations

Section 6 of RFC 6712 [RFC6712] contains the IANA Considerations of that document. As this document defines a new well-known URI, the IANA Considerations need to be updated accordingly.

Add the following text between the first and second paragraph of the section.

Within the well-known URI registry (see https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml#well-known-uris-1) as defined in RFC 8515 [RFC8515] the following change has been performed.

One new name entry has been added:

URI suffix	Change controller
cmp	IETF

4. IANA Considerations

This document contains an update to the IANA Consideration sections to be added to [RFC4210] and [RFC6712].

< TBD: This document updates the ASN.1 modules of RFC 4210 Appendix F [RFC4210] and RFC 5912 Section 9 [RFC5912]. New OIDs TBD1 and TBD2 need to be registered to identify the updates ASN.1 modules. >

< TBD: New OIDs TBD3 (id-regCtrl-algId) and TBD4 (id-regCtrl-rsaKeyLen) need to be registered. >

< TBD: The existing description and information of id-kp-cmcRA and id-kp-cmcCA need to be updated to reflect their extended usage. >

5. Security Considerations

No changes are made to the existing security considerations of RFC 4210 [RFC4210] and RFC 6712 [RFC6712].

6. Acknowledgements

Special thank goes to Jim Schaad for his guidance and the inspiration on structuring and writing this document I got from [RFC6402] that updates CMC. Special thank also goes also to Russ Housley and Tomas Gustavsson for reviewing and providing valuable suggestions on the approvement of this document.

I also like to thank all reviewers of this document for their valuable feedback.

7. References

7.1. Normative References

[I-D.ietf-lamps-cmp-algorithms]
Brockhaus, H., "CMP Algorithms", draft-ietf-lamps-cmp-algorithms-00 (work in progress), October 2020.

[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>.

[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>.

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

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

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

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

[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>.

[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>.

[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>.

[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>.

[RFC8515] Jethanandani, M. and M. Reina Ortega, "URN Namespace for ETSI Documents", RFC 8515, DOI 10.17487/RFC8515, February 2019, <https://www.rfc-editor.org/info/rfc8515>.

7.2. Informative References

[I-D.ietf-lamps-lightweight-cmp-profile]
Brockhaus, H., Fries, S., and D. Oheimb, "Lightweight CMP Profile", draft-ietf-lamps-lightweight-cmp-profile-03 (work in progress), October 2020.

[IEEE802.1AR]
IEEE, "802.1AR Secure Device Identifier", June 2018, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

Appendix A. ASN.1 Modules

A.1. 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 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-cmp2020-88(TBD1)}

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL --

IMPORTS

    Certificate, CertificateList, Extensions, AlgorithmIdentifier,
    UTF8String, id-kp -- 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(1)}

    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(2)}
```

```
      CertTemplate, PKIPublicationInfo, EncryptedKey, CertId,
      CertReqMessages, Controls, 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 [thisRFC]]. 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) member-body(2)
                           us(840) rsadsi(113549)
                           pkcs(1) pkcs-10(10) modules(1) pkcs-10(1)}
      -- (specified in RFC 2986 with 1993 ASN.1 syntax and IMPLICIT
      -- tags).  Alternatively, implementers may directly include
      -- the [PKCS10] syntax 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 [thisRFC]

      EnvelopedData, SignedData
             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 [thisRFC]

      ;

   -- 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, WAP WTLS 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 "un-comment" 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) },
       sender              GeneralName,
       -- identifies the sender
       recipient           GeneralName,
       -- identifies the intended recipient
       messageTime     [0] GeneralizedTime         OPTIONAL,
       -- time of production of this message (used when 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 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 UTF-8 String [RFC3629] (note: each
       -- UTF8String MAY include an [RFC3066] language tag
       -- to indicate the language of the contained text
       -- see [RFC2482] for details)

   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 [PKCS10]
       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 (SHA-1 recommended)
       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., DES-MAC, Triple-DES-MAC [PKCS11],
   }   -- 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 (SHA-1 recommended)
       mac                 AlgorithmIdentifier
       -- the MAC AlgId (e.g., DES-MAC, Triple-DES-MAC [PKCS11],
   }   -- 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; 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),
           -- invalid integrity, password based instead of signature or
           -- vice versa
        badRecipientNonce   (13),
           -- invalid 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),
```

```
          -- invalid sender nonce, either missing or wrong size
       badCertTemplate     (19),
          -- invalid 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)
       -- 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.
    }

    -- Added in CMP Updates [thisRFC]

    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 corresponding request (a value
        -- of -1 is to be used if certReqId is not specified in the
        -- corresponding request)
        status              PKIStatusInfo,
        certifiedKeyPair    CertifiedKeyPair    OPTIONAL,
        rspInfo             OCTET STRING        OPTIONAL
        -- analogous to the id-regInfo-utf8Pairs string defined
        -- for regInfo in CertReqMsg [CRMF]
    }

    CertifiedKeyPair ::= SEQUENCE {
        certOrEncCert       CertOrEncCert,
        privateKey      [0] EncryptedKey        OPTIONAL,
        -- see [CRMF] for comment on encoding
        -- Changed from Encrypted Value to EncryptedKey as a CHOICE of
```

```
        -- EncryptedValue and EnvelopedData due to the changes made in
        -- CMP Updates [thisRFC]
        -- 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 [thisRFC]
        -- 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 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 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
   }

   PKIConfirmContent ::= NULL

   -- Added in CMP Updates [thisRFC]

   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
      }

   -- Added in CMP Updates [thisRFC]

   CertReqTemplateContent ::= SEQUENCE {
```

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

      id-regCtrl-algId OBJECT IDENTIFIER ::= { id-regCtrl TBD3 }
      AlgIdCtrl ::= AlgorithmIdentifier
      -- SHALL be used to specify suported algorithms other than RSA

      id-regCtrl-rsaKeyLen OBJECT IDENTIFIER ::= { id-regCtrl TBD4 }
      RsaKeyLenCtrl ::= Integer
      -- SHALL be used to specify suported RSA key lengths

   InfoTypeAndValue ::= SEQUENCE {
       infoType               OBJECT IDENTIFIER,
       infoValue              ANY DEFINED BY infoType  OPTIONAL
   }
   -- Example InfoTypeAndValue contents include, but are not limited
   -- to, the following (un-comment 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 OF AlgorithmIdentifier
   --   id-it-encKeyPairTypes  OBJECT IDENTIFIER ::= {id-it 3}
   --      EncKeyPairTypesValue    ::= SEQUENCE 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 OF OBJECT IDENTIFIER
   --   id-it-keyPairParamReq  OBJECT IDENTIFIER ::= {id-it 10}
   --      KeyPairParamReqValue    ::= OBJECT IDENTIFIER
   --   id-it-keyPairParamRep  OBJECT IDENTIFIER ::= {id-it 11}
   --      KeyPairParamRepValue    ::= AlgorithmIdentifer
   --   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 [thisRFC]
   --      -- 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 OF CMPCertificate
   --      -- id-it-caCerts added in CMP Updates [thisRFC]
   --   id-it-rootCaKeyUpdate OBJECT IDENTIFIER ::= { id-it 18}
   --      RootCaKeyUpdateValue ::= RootCaKeyUpdateContent
   --      -- id-it-rootCaKeyUpdate added in CMP Updates [thisRFC]
   --   id-it-certReqTemplate OBJECT IDENTIFIER ::= { id-it 19}
   --      CertReqTemplateValue ::= CertReqTemplateContent
   --      -- id-it-certReqTemplate added in CMP Updates [thisRFC]
   --
   -- 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 OBJ. IDs 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
   -- 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 [thisRFC]
   -- 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 -- of CMP module
```

A.2. 2002 ASN.1 Module

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

< TBD: Dose this document then also updates [RFC5912]? >

```
   PKIXCMP-2009
       { iso(1) identified-organization(3) dod(6) internet(1)
       security(5) mechanisms(5) pkix(7) id-mod(0)
       id-mod-cmp2020-02(TBD2) }
   DEFINITIONS EXPLICIT TAGS ::=
   BEGIN
   IMPORTS
```

```
   AttributeSet{}, 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, 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)}

   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, 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 [thisRFC]. 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 RFC 2986 with 1993 ASN.1 syntax and IMPLICIT
   -- tags).  Alternatively, implementers may directly include
   -- the [PKCS10] syntax 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 [thisRFC]

  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 [thisRFC]
  ;

  -- 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, WAP WTLS 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) },
      sender              GeneralName,
      -- identifies the sender
      recipient           GeneralName,
      -- identifies the intended recipient
      messageTime     [0] GeneralizedTime         OPTIONAL,
      -- time of production of this message (used when 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 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 UTF-8 String [RFC3629] (note: each
      -- UTF8String MAY include an [RFC3066] language tag
      -- to indicate the language of the contained text;
      -- see [RFC2482] for details)

   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 [PKCS10]
      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 (SHA-1 recommended)
        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., DES-MAC, Triple-DES-MAC [PKCS11],
        -- 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 (SHA-1 recommended)
        mac                 AlgorithmIdentifier{MAC-ALGORITHM, {...}}
        -- the MAC AlgId (e.g., DES-MAC, Triple-DES-MAC [PKCS11],
        -- 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; 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),
        -- invalid integrity, password based instead of signature or
        -- vice versa
        badRecipientNonce   (13),
        -- invalid 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),
        -- invalid sender nonce, either missing or wrong size
        badCertTemplate     (19),
        -- invalid 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)
        -- 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.
   }

   -- Added in CMP Updates [thisRFC]

   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)
       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 comment on encoding
       -- Changed from Encrypted Value to EncryptedKey as a CHOICE of
       -- EncryptedValue and EnvelopedData due to the changes made in
       -- CMP Updates [thisRFC]
       -- 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 [thisRFC]
       -- 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 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 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

    -- Added in CMP Updates [thisRFC]

    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
      }

   -- Added in CMP Updates [thisRFC]

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

   id-regCtrl-algId OBJECT IDENTIFIER ::= { id-regCtrl TBD3 }
   AlgIdCtrl ::= AlgorithmIdentifier
   -- SHALL be used to specify suported algorithms other than RSA

   id-regCtrl-rsaKeyLen OBJECT IDENTIFIER ::= { id-regCtrl TBD4 }
   RsaKeyLenCtrl ::= Integer
   -- SHALL be used to specify suported RSA key lengths

   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 OF
   --                                  AlgorithmIdentifier{{...}}
   --   id-it-encKeyPairTypes  OBJECT IDENTIFIER ::= {id-it 3}
   --      EncKeyPairTypesValue    ::= SEQUENCE 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 OF OBJECT IDENTIFIER
   --   id-it-keyPairParamReq  OBJECT IDENTIFIER ::= {id-it 10}
   --      KeyPairParamReqValue    ::= OBJECT IDENTIFIER
   --   id-it-keyPairParamRep  OBJECT IDENTIFIER ::= {id-it 11}
   --      KeyPairParamRepValue    ::= AlgorithmIdentifer
   --   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 [thisRFC]
   --      -- 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 OF CMPCertificate
   --      -- id-it-caCerts added in CMP Updates [thisRFC]
   --   id-it-rootCaKeyUpdate OBJECT IDENTIFIER ::= { id-it 18}
   --      RootCaKeyUpdateValue ::= RootCaKeyUpdateContent
   --      -- id-it-rootCaKeyUpdate added in CMP Updates [thisRFC]
   --   id-it-certReqTemplate OBJECT IDENTIFIER ::= { id-it 19}
   --      CertReqTemplateValue ::= CertReqTemplateContent
   --      -- id-it-certReqTemplate added in CMP Updates [thisRFC]
   --
   -- 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 OBJECT IDs 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
   -- 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 }

   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 [thisRFC]
   -- 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
```

Appendix B. History of changes

Note: This appendix will be deleted in the final version of the document.

From version 05 -> 06:

- Added the update of Appendix D.2 with the reference to the new CMP Algorithms document as decided in IETF 108
- Updated the IANA considerations to register new OIDs for id-regCtrl-algId and d-regCtrl-rsaKeyLen.
- Minor changes and corrections

From version 04 -> 05:

- Added Section 2.6 and Section 2.7 to clarify the usage of these general messages types with EC curves (see thread "AlgorithmIdentifier parameters NULL value - Re: InfoTypeAndValue in CMP headers")
- Split former section 2.7 on adding 'CA Certificates', 'Root CA Certificates Update', and 'Certificate Request Template' in three separate sections for easier readability
- Changed in Section 2.10 the ASN.1 syntax of CertReqTemplateValue from using reaKeyLen to usage of controls as specified in CRMF Section 6 [RFC4211] (see thread "dtaft-ietf-lamps-cmp-updates and rsaKeyLen")
- Updated the IANA considerations in Section 2.13 to introduce new OID for id-regCtrl-algId and id-regCtrl-rsaKeyLen (see thread "dtaft-ietf-lamps-cmp-updates and rsaKeyLen")
- Updated the IANA Considerations in and the Appendixes to introduce new OID for the updates ASN.1 modules (see thread "I-D Action: draft-ietf-lamps-cmp-updates-04.txt")

- Removed EncryptedValue from and added Controls to the list of types imported from CRMF [RFC4211] in ASN.1 modules (see thread "draft-ietf-lamps-cmp-updates and the ASN.1 modules")

- Moved declaration of Rand out of the comment in ASN.1 modules (see thread "draft-ietf-lamps-cmp-updates and the ASN.1 modules")

- Minor changes and corrections

From version 03 -> 04:

- Added Section 2.7 to introduce three new id-it IDs for uses in general messages as discussed (see thread "draft-ietf-lamps-cmp-updates add section to introduce id-it-caCerts, id-it-rootCaKeyUpdate, and id-it-certReqTemplate")

- Added the new id-it IDs and the /.well-known/cmp to the IANA Considerations of [RFC4210] in Section 2.9

- Updated the IANA Considerations of [RFC4210] in Section 2.14

- Some changes in wording on Section 3 due to review comments from Martin Peylo

From version 02 -> 03:

- Added a ToDo on aligning with the CMP Algorithms draft that will be set up as decided in IETF 108

- Updated section on Encrypted Values in Section 2.4 to add the AsymmetricKey Package structure to transport a newly generated private key as decided in IETF 108

- Updated the IANA Considerations of [RFC4210] in Section 2.14

- Added the pre-registered OID in Section 2.14 and the ASN.1 module

- Added Section 3 to document the changes to RFC 6712 [RFC6712] regarding URI discovery and using the path-prefix of '/.well-known/' as discussed in IETF 108

- Updated the IANA Considerations section

- Added a complete updated ASN.1 module in 1988 syntax to update Appendix F of [RFC4210] and a complete updated ASN.1 module in 2002 syntax to update Section 9 of [RFC5912]

- Minor changes in wording

From version 01 -> 02:

- Updated section on EKU OIDs in Section 2.2 as decided in IETF 107
- Changed from symmetric key-encryption to password-based key management technique in Section 2.4 as discussed with Russ and Jim on the mailing list
- Defined the attribute containing the key identifier for the revocation passphrase in Section 2.14
- Moved the change history to the Appendix

From version 00 -> 01:

- Minor changes in wording

From draft-brockhaus-lamps-cmp-updates-03 -> draft-ietf-lamps-cmp-updates-00:

- Changes required to reflect WG adoption

From version 02 -> 03:

- Added some clarification in Section 2.1

From version 01 -> 02:

- Added clarification to section on multiple protection
- Added clarification on new EKUs after some exchange with Tomas Gustavsson
- Reused OIDs from RFC 6402 [RFC6402] as suggested by Sean Turner at IETF 106
- Added clarification on the field containing the key identifier for a revocation passphrase
- Minor changes in wording

From version 00 -> 01:

- Added a section describing the new extended key usages
- Completed the section on changes to the specification of encrypted values

- Added a section on clarification to Appendix D.4
- Minor generalization in RFC 4210 [RFC4210] Sections 5.1.3.4 and 5.3.22
- Minor changes in wording

Author's Address

Hendrik Brockhaus
Siemens AG

Email: hendrik.brockhaus@siemens.com

LAMPS Working Group
Internet-Draft
Intended status: Standards Track
Expires: 7 May 2021

B. Hoeneisen
pEp Foundation
D.K. Gillmor
American Civil Liberties Union
A. Melnikov
Isode Ltd

3 November 2020

Header Protection for S/MIME
draft-ietf-lamps-header-protection-02

Abstract

S/MIME version 3.1 has introduced a feasible standardized option to accomplish Header Protection. However, implementations of Header Protection can cause rendering issues on the receiving side. Clearer specifications regarding message processing, particularly with respect to header sections, are needed in order to resolve these rendering issues.

In order to help implementers to correctly compose and render email messages with Header Protection, this document updates S/MIME Header Protection specifications with additional guidance on MIME format, sender and receiver processing.

Table of Contents

1. Introduction

Privacy and security issues regarding email Header Protection in S/MIME have been identified for some time. Most current implementations of cryptographically-protected electronic mail protect only the body of the message, which leaves significant room for attacks against otherwise-protected messages. For example, lack of header protection allows an attacker to substitute the message subject and/or author.

A way to provide end-to-end protection for the Header Section of an email message has been standardized for S/MIME version 3.1 and later (cf. [RFC8551]):

> In order to protect outer, non-content-related message header fields (for instance, the "Subject", "To", "From", and "Cc" fields), the sending client MAY wrap a full MIME message in a message/RFC822 wrapper in order to apply S/MIME security services to these header fields.

Unfortunately, implementations of Header Protection can cause rendering issues on the receiving side. In some cases, the user sees an attachment suggesting a forwarded email message, which - in fact - contains the protected email message that should be rendered directly. For these cases, the user can click on the attachment to view the protected message. However, there have also been reports of email clients displaying garbled text, or sometimes nothing at all. In those cases the email clients on the receiving side are (most likely) not fully MIME-capable.

The following shortcomings have been identified to cause these issues:

* Broken or incomplete implementations
* Lack of a simple means to distinguish "forwarded message" and "wrapped message" (for the sake of Header Protection)
* Not enough guidance with respect to handling of Header Fields on both the sending and the receiving side

Furthermore, the need (technical) Data Minimization, which includes data sparseness and hiding all technically concealable information, has grown in importance over the past several years. In addition,

backwards compatibility must be considered when it is possible to do so without compromising privacy and security.

No mechanism for Header Protection has been standardized for PGP/MIME (Pretty Good Privacy) [RFC3156] yet. PGP/MIME developers have implemented ad-hoc header-protection, and would like to see a specification that is applicable to both S/MIME and PGP/MIME.

This document describes the problem statement (Section 2), generic use cases (Section 3) and the specification for Header Protection (Section 4) with guidance on MIME format, sender and receiver processing .

[I-D.ietf-lamps-header-protection-requirements] defines the requirements that this specification is based on.

This document is in an early draft state and contains a proposal on which to base future discussions of this topic. In any case, the final mechanism is to be determined by the IETF LAMPS WG.

1.1. Other Protocols to Protect Email Headers

A range of protocols for the protection of electronic mail (email) exists, which allows one to assess the authenticity and integrity of the email headers section or selected Header Fields from the domain-level perspective, specifically DomainKeys Identified Mail (DKIM) However, integrity protection and proof of authenticity are both tied to the domain name of the sending e-mail address, not the sending address itself, so these protocols do not provide end-to-end protection, and are incapable of providing any form of confidentiality.[RFC6376], as used by Domain-based Message Authentication, Reporting, and Conformance (DMARC) [RFC7489]. These protocols provide a domain-based reputation mechanism that can be used to mitigate some forms of unsolicited email (spam). At the same time, these protocols can provide a level of cryptographic integrity and authenticity for some headers, depending on how they are used.

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.3. Terms

The following terms are defined for the scope of this document:

* Man-in-the-middle (MITM) attack: cf. [RFC4949], which states: "A form of active wiretapping attack in which the attacker intercepts and selectively modifies communicated data to masquerade as one or more of the entities involved in a communication association."

  Note: Historically, MITM has stood for '_Man_-in-the-middle'. However, to indicate that the entity in the middle is not always a human attacker, MITM can also stand for 'Machine-in-the-middle' or 'Meddler-in-the-middle'.

* S/MIME: Secure/Multipurpose Internet Mail Extensions (cf. [RFC8551])

* PGP/MIME: MIME Security with OpenPGP (cf. [RFC3156])

* Message: An Email Message consisting of Header Fields (collectively called "the Header Section of the message") followed, optionally, by a Body; cf. [RFC5322].

  Note: To avoid ambiguity, this document does not use the terms "Header" or "Headers" in isolation, but instead always uses "Header Field" to refer to the individual field and "Header Section" to refer to the entire collection; cf. [RFC5322].

* Header Field (HF): cf. [RFC5322] Header Fields are lines beginning with a field name, followed by a colon (":"), followed by a field body (value), and terminated by CRLF.

* Header Section (HS): The Header Section is a sequence of lines of characters with special syntax as defined in [RFC5322]. It is the (top) section of a Message containing the Header Fields.

* Body: The Body is simply a sequence of bytes that follows the Header Section and is separated from the Header Section by an empty line (i.e., a line with nothing preceding the CRLF); cf [RFC5322]. It is the (bottom) section of Message containing the payload of a Message. Typically, the Body consists of a (possibly multipart) MIME [RFC2045] construct.

* MIME Header Fields: Header Fields describing content of a MIME entity [RFC2045], in particular the MIME structure. Each MIME Header Field name starts with "Content-" prefix.

* MIME Header Section (part): The collection of MIME Header Fields. "MIME Header Section" refers to a Header Sections that contains only MIME Header Fields, whereas "MIME Header Section part" refers to the MIME Header Fields of a Header Section that - in addition to MIME Header Fields - also contains non-MIME Header Fields.

* Essential Header Fields (EHF): The minimum set of Header Fields an Outer Message Header Section SHOULD contain; cf. Section 4.1.2.4.

* Header Protection (HP): cryptographic protection of email Header Sections (or parts of it) for signatures and/or encryption

* Protection Levels (PL): The level of protection applied to a Message, e.g. 'signature and encryption' or 'signature only' (cf. Section 3.2).

* Protected: Portions of a message that have had any Protection Levels applied.

* Protected Message: A Message that has had any Protection Levels applied.

* Unprotected: Portions of a Message that has had no Protection Levels applied.

* Unprotected Message: A Message that has had no Protection Levels applied.

* Submission Entity: The entity which executes further processing of the Message (incl. transport towards the receiver), after protection measures have been applied to the Message.

  Note: The Submission Entity varies among implementations, mainly depending on the stage where protection measures are applied: E.g. a Message Submission Agent (MSA) [RFC6409] or another (proprietary) solution. The latter is particularly relevant, if protection is implemented as a plugin solution. Some implementations may determine the destination recipients by reading the To, Cc and Bcc Header Fields of the Outer Message.

* Original Message (OrigM): The Message to be protected before any protection-related processing has been applied on the sending side. If the source is not a "message/rfc822" Message, OrigM is defined as the "virtual" Message that would be constructed for sending it as unprotected email.

* Inner Message (InnerM): The Message to be protected which has had wrapping and protection measures aapplied on the sending side OR the resulting Message once decryption and unwrapping on the receiving side has been performed. Typically, the Inner Message is in clear text. The Inner Message is a subset of (or the same as) the Original Message (cf. Section 4.1.2.1). The Inner Message must be the same on the sending and the receiving side.

* Outer Message (OuterM): The Message as provided to the Submission Entity or received from the last hop respectively. The Outer Message normally differs on the sending and the receiving side (e.g. new Header Fields are added by intermediary nodes).

* Receiving User Facing Message (RUFM): The Message used for rendering at the receiving side. Typically this is the same as the Inner Message.

* Data Minimization: Data sparseness and hiding of all technically concealable information whenever possible.

* Cryptographic Layer, Cryptographic Payload, and Cryptographic Envelope are all used as defined in [I-D.dkg-lamps-e2e-mail-guidance]

2. Problem Statement

The LAMPS charter contains the following Work Item:

> Update the specification for the cryptographic protection of email headers - both for signatures and encryption - to improve the implementation situation with respect to privacy, security, usability and interoperability in cryptographically-protected electronic mail. Most current implementations of cryptographically-protected electronic mail protect only the body of the message, which leaves significant room for attacks against otherwise-protected messages.

In the following a set of challenges to be addressed:

[[TODO: Enhance this section, add more items to the following.]]

2.1. Privacy

* (Technical) Data Minimization, which includes data sparseness and hiding all technically concealable information whenever possible

2.2. Security

* Prevent MITM attacks (cf. [RFC4949])

2.3. Usability

* Improved User interaction / User experience, in particular at the receiving side

2.4. Interoperability

* Interoperability with [RFC8551] implementations

3. Use Cases

In the following, the reader can find a list of the generic use cases that need to be addressed for Messages with Header Protection (HP). These use cases apply regardless of technology (S/MIME, PGP/MIME, etc.) used to achieve HP.

3.1. Interactions

The following use cases assume that at least the sending side supports Header Protection as specified in this document. Receiving sides that support this specification are expected to be able to distinguish between Messages that use Header Protection as specified in this document, and (legacy) Mail User Agents (MUAs) which do not implement this specification.

[[TODO: Verify once solution is stable and update last sentence.]]

3.1.1. Main Use Case

Both the sending and receiving side (fully) support Header Protection as specified in this document.

The main use case is specified in Section 4.1.

3.1.2. Backward Compatibility Use Cases

Regarding backward compatibility, the main distinction is based on whether or not the receiving side conforms to MIME according to [RFC2046], ff., which in particular also includes Section 2 of [RFC2049] on "MIME Conformance". The following excerpt is contextually relevant:

> A mail user agent that is MIME-conformant MUST:
>
> [...]
>
> -- Recognize and display at least the RFC822 message encapsulation (message/rfc822) in such a way as to preserve any recursive structure, that is, displaying or offering to display the encapsulated data in accordance with its media type.
>
> -- Treat any unrecognized subtypes as if they were "application/octet-stream".
>
> [...]
>
> An MUA that meets the above conditions is said to be MIME-conformant. A MIME-conformant MUA is assumed to be "safe" to send virtually any kind of properly-marked data to users of such mail systems, because these systems are, at a minimum, capable of treating the data as undifferentiated binary, and will not simply splash it onto the screen of unsuspecting users.

[[TODO: The compatibility of legacy HP systems with this new solution, and how to handle issues surrounding future maintenance for these legacy systems, will be decided by the LAMPS WG.]]

3.1.2.1. Receiving Side MIME-Conformant

The sending side (fully) supports Header Protection as specified in this document, while the receiving side does not support this specification. However, the receiving side is MIME-conformant according to [RFC2045], ff. (cf. Section 3.1.2).

This use case is specified in Section 4.2.1.

Note: This case should perform as expected if the sending side applies this specification as outlined in Section 4.1.

[[TODO: Verify once solution is stable and update last sentence.]]

3.1.2.2. Receiving Side Not MIME-Conformant

The sending side (fully) supports Header Protection as specified in this document, while the receiving side does not support this specification. Furthermore, the receiving side is *not* MIME-conformant according to [RFC2045], ff. (cf. Section 3.1.2).

This use case is specified in Section 4.2.2.

3.2. Protection Levels

3.2.1. In-Scope

The following Protection Levels are in scope for this document:

a) Signature and encryption

Messages containing a cryptographic signature, which are also encrypted.

b) Signature only

Messages containing a cryptographic signature, but which are not encrypted.

3.2.2. Out-of-Scope

Legacy implementations, implementations not (fully) compliant with this document or corner-cases may lead to further Protection Levels to appear on the receiving side, such as (list not exhaustive):

* Triple wrap
* Encryption only
* Encryption before signature
* Signature and encryption, but:
  - Signature fails to validate
  - Signature validates but the signing certificate revoked
* Signature only, but:
  - with multiple valid signatures, layered atop each other

These Protection Levels, as well as any further Protection Levels not listed in Section 3.2.1 are beyond the scope of this document.

4. Specification

This section contains the specification for Header Protection in S/MIME to update and clarify Section 3.1 of [RFC8551] (S/MIME 4.0).

Note: It is likely that PGP/MIME [RFC3156] will also incorporate this specification or parts of it.

This specification applies to the Protection Levels "signature & encryption" and "signature only" (cf. Section 3.2):

Sending and receiving sides MUST implement the "signature and encryption" Protection Level, which SHOULD be used as default on the sending side.

Certain implementations may decide to send "signature only" Messages, depending on the circumstances and customer requirements. Sending sides MAY and receiving sides MUST implement "signature only" Protection Level.

It generally is NOT RECOMMENDED to send a Message with any other Protection Level. On the other hand, the receiving side must be prepared to receive Messages with other Protection Levels.

[[TODO: Further study is necessary to determine whether - and if yes to what extent - additional guidance for handling messages with other Protection Levels, e.g. "encryption only" at the receiving side should be included in this document.]]

4.1. Main Use Case

This section applies to the main use case, where the sending and receiving side (fully) support Header Protection as specified herein (cf. Section 3.1.1).

Note: The sending side specification of the main use case is also applicable to the cases where the sending side (fully) supports Header Protection as specified herein, while the receiving side does not, but is MIME-conformant according to [RFC2045], ff. (cf. Section 3.1.2 and Section 3.1.2.1).

Further backward compatibility cases are defined in Section 4.2.

4.1.1. MIME Format

4.1.1.1. Introduction

As per S/MIME version 3.1 and later (cf. [RFC8551]), the sending client MAY wrap a full MIME message in a message/RFC822 wrapper in order to apply S/MIME security services to these header fields.

To help the receiving side to distinguish between a forwarded and a wrapped message, the Content-Type header field parameter "forwarded" is added as defined in [I-D.melnikov-iana-reg-forwarded].

The simplified (cryptographic overhead not shown) MIME structure of such an Email Message looks as follows:

```
<Outer Message Header Section (unprotected)>

<Outer Message Body (protected)>

  <MIME Header Section (wrapper)>

    <Inner Message Header Section>

    <Inner Message Body>
```

The following example demonstrates how an Original Message might be protected, i.e., the Original Message is contained as Inner Message in the Protected Body of an Outer Message. It illustrates the first Body part (of the Outer Message) as a "multipart/signed" (application/pkcs7-signature) media type:

Lines are prepended as follows:

* "O: " Outer Message Header Section

* "I: " Message Header Section

* "W: " Wrapper (MIME Header Section)

```
O: Date: Mon, 25 Sep 2017 17:31:42 +0100 (GMT Daylight Time)
O: Message-ID: <e4a483cb-1dfb-481d-903b-298c92c21f5e@m.example.net>
O: Subject: Meeting at my place
O: From: "Alexey Melnikov" <alexey.melnikov@example.net>
O: To: somebody@example.net
O: MIME-Version: 1.0
O: Content-Type: multipart/signed; charset=us-ascii; micalg=sha1;
O:  protocol="application/pkcs7-signature";
O:  boundary=boundary-AM

   This is a multipart message in MIME format.
   --boundary-AM
W: Content-Type: message/RFC822; forwarded=no
W:
I: Date: Mon, 25 Sep 2017 17:31:42 +0100 (GMT Daylight Time)
I: From: "Alexey Melnikov" <alexey.melnikov@example.net>
I: Message-ID: <e4a483cb-1dfb-481d-903b-298c92c21f5e@m.example.net>
I: MIME-Version: 1.0
I: MMHS-Primary-Precedence: 3
I: Subject: Meeting at my place
I: To: somebody@example.net
I: X-Mailer: Isode Harrier Web Server
I: Content-Type: text/plain; charset=us-ascii

   This is an important message that I don't want to be modified.

   --boundary-AM
   Content-Transfer-Encoding: base64
   Content-Type: application/pkcs7-signature

   [[base-64 encoded signature]]

   --boundary-AM--
```

The Outer Message Header Section is unprotected, while the remainder (Outer Message Body) is protected. The Outer Message Body consists of the wrapper (MIME Header Section) and the Inner Message (Header Section and Body).

The wrapper is a simple MIME Header Section with media type "message/rfc822" containing a Content-Type header field parameter "forwarded=no" followed by an empty line.

If the source is an Original (message/rfc822) Message, the Inner Message Header Section is typically the same as (or a subset of) the Original Message Header Section (cf. Section 4.1.2.1), and the Inner Message Body is typically the same as the Original Message Body.

The Inner Message itself may contain any MIME structure.

Note: It is still to be decided by the LAMPS WG whether or not to recommend an alternative MIME format as described in Appendix B.1.1.1 (instead of the currently standardized and above defined format).

4.1.2. Sending Side

To ease explanation, the following describes the case where an Original (message/rfc822) Message to be protected is present. If this is not the case, Original Message means the (virtual) Message that would be constructed for sending it as unprotected email.

4.1.2.1. Inner Message Header Fields

It is RECOMMENDED that the Inner Message contains all Header Fields of the Original Message with the exception of the following Header Field, which MUST NOT be included within the Inner Message nor within any other protected part of the Message:

* Bcc

[[TODO: Bcc handling needs to be further specified (see also Appendix A.1). Certain MUAs cannot properly decrypt Messages with Bcc recipients.]]

4.1.2.2. Wrapper

The wrapper is a simple MIME Header Section followed by an empty line preceding the Inner Message (inside the Outer Message Body). The media type of the wrapper MUST be "message/RFC822" and MUST contain the Content-Type header field parameter "forwarded=no" as defined in [I-D.melnikov-iana-reg-forwarded]. The wrapper unambiguously delimits the Inner Message from the rest of the Message.

4.1.2.3. Cryptographic Layers / Envelope

[[TODO: Basically refer to S/MIME standards]]

4.1.2.4. Outer Message Header Fields

4.1.2.4.1. Encrypted Messages

To maximize Privacy, it is strongly RECOMMENDED to follow the principle of Data Minimization (cf. Section 2.1).

However, the Outer Message Header Section SHOULD contain the Essential Header Fields and, in addition, MUST contain the Header

Fields of the MIME Header Section part to describe Cryptographic Layer of the protected MIME subtree as per [RFC8551].

The following Header Fields are defined as the Essential Header Fields:

* From

* To (if present in the Original Message)

* Cc (if present in the Original Message)

* Bcc (if present in the Original Message, see also Section 4.1.2.1 and Appendix A.1)

* Date

* Message-ID

* Subject

Further processing by the Submission Entity normally depends on part of these Header Fields, e.g. From and Date HFs are required by [RFC5322]. Furthermore, not including certain Header Fields may trigger spam detection to flag the Message, and/or lead to user experience (UX) issues.

For further Data Minimization, the value of the Subject Header Field SHOULD be obfuscated as follows:

* Subject: [...]

and it is RECOMMENDED to replace the Message-ID by a new randomly generated Message-ID.

In addition, the value of other Essential Header Fields MAY be obfuscated.

Non-Essential Header Fields SHOULD be omitted from the Outer Message Header Section where possible. If Non-essential Header Fields are included in the Outer Message Header Section, those MAY be obfuscated too.

Header Fields that are not obfuscated should contain the same values as in the Original Message.

If an implementation obfuscates the From, To, and/or Cc Header Fields, it may need to provide access to the clear text content of

these Header Fields to the Submission Entity for processing purposes. This is particularly relevant, if proprietary Submission Entities are used. Obfuscation of Header Fields may adversely impact spam filtering.

(A use case for obfuscation of all Outer Message Header Fields is routing email through the use of onion routing or mix networks, e.g. [pEp.mixnet].)

The MIME Header Section part is the collection of MIME Header Fields describing the following MIME structure as defined in [RFC2045]. A MIME Header Section part typically includes the following Header Fields:

* Content-Type

* Content-Transfer-Encoding

* Content-Disposition

The following example shows the MIME Header Section part of an S/MIME signed Message (using application/pkcs7-mime with SignedData):

```
MIME-Version: 1.0
Content-Type: application/pkcs7-mime; smime-type=signed-data;
   name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
```

Depending on the scenario, further Header Fields MAY be exposed in the Outer Message Header Section, which is NOT RECOMMENDED unless justified. Such Header Fields may include e.g.:

* References

* Reply-To

* In-Reply-To

4.1.2.4.2. Unencrypted Messages

The Outer Message Header Section of unencrypted Messages SHOULD contain at least the Essential Header Fields and, in addition, MUST contain the Header Fields of the MIME Header Section part to describe Cryptographic Layer of the protected MIME subtree as per [RFC8551]. It may contain further Header Fields, in particular those also present in the Inner Message Header Section.

4.1.2.5. Sending Side Message Processing

For a protected Message the following steps are applied before a Message is handed over to the Submission Entity:

4.1.2.5.1. Step 1: Decide on Protection Level and Information Disclosure

The implementation which applies protection to a Message must decide:

* Which Protection Level (signature and/or encryption) shall be applied to the Message? This depends on user request and/or local policy as well as availability of cryptographic keys.

* Which Header Fields of the Original Message shall be part of the Outer Message Header Section? This typically depends on local policy. By default, the Essential Header Fields are part of the Outer Message Header Section; cf. Section 4.1.2.4.

* Which of these Header Fields are to be obfuscated? This depends on local policy and/or specific Privacy requirements of the user. By default only the Subject Header Field is obfuscated; cf. Section 4.1.2.4.

4.1.2.5.2. Step 2: Compose the Outer Message Header Section

Depending on the decision in Section 4.1.2.5.1, the implementation shall compose the Outer Message Header Section. (Note that this also includes the necessary MIME Header Section part for the following protection layer.)

Outer Header Fields that are not obfuscated should contain the same values as in the Original Message (except for MIME Header Section part, which depends on the Protection Level selected in Section 4.1.2.5.1).

4.1.2.5.3. Step 3: Apply Protection to the Original Message

Depending on the Protection Level selected in Section 4.1.2.5.1, the implementation applies signature and/or encryption to the Original Message, including the wrapper (as per [RFC8551]), and sets the resulting package as the Outer Message Body.

The resulting (Outer) Message is then typically handed over to the Submission Entity.

[[TODO: Example]]

4.1.3. Receiving Side

4.1.3.1. Receiving User Facing Message Header Fields

The Receiving User Facing Message SHOULD be a verbatim copy of the Inner Message.

4.1.3.2. Receiving Side Message Processing

When a protected Message is received, the following steps are applied:

4.1.3.2.1. Step 1: Decrypt Message and/or check signature

Depending on the Protection Level, the received Message is decrypted and/or its signature is checked as per [RFC8551].

4.1.3.2.2. Step 2: Construct the Receiving User Facing Message

The Receiving User Facing Message is constructed according to Section 4.1.3.1.

The resulting Message is handed over for further processing, which typically involves rendering it for the user.

4.1.3.3. Step 3: Prepare Information Cyptographic Verification

[[TODO: Signature valid, etc.]]

4.2. Backward Compatibility Use Cases

4.2.1. Receiving Side MIME-Conformant

This section applies to the case where the sending side (fully) supports Header Protection as specified in this document, while the receiving side does not support this specification, but is MIME-conformant according to [RFC2045], ff. (cf. Section 3.1.2 and Section 3.1.2.1)

The sending side specification of the main use case (cf. Section 4.1) MUST ensure that receiving sides can still recognize and display or offer to display the encapsulated data in accordance with its media type (cf. [RFC2049], Section 2). In particular, receiving sides that do not support this specification, but are MIME-conformant according to [RFC2045], ff. can still recognize and display the Message intended for the user.

[[TODO: Verify once solution is stable and update last sentence.]]

4.2.2. Receiving Side Not MIME-Conformant

This section applies to cases where the sending side (fully) supports Header Protection as specified in this document, while the receiving side neither supports this specification *nor* is MIME-conformant according to [RFC2045], ff. (cf. Section 3.1.2 and Section 3.1.2.2).

[I-D.autocrypt-lamps-protected-headers] describes a possible way to achieve backward compatibility with existing S/MIME (and PGP/MIME) implementations that predate this specification and are not MIME-conformant (Legacy Display) either. It mainly focuses on email clients that do not render emails which utilize header protection in a user friendly manner, which may confuse the user. While this has been observed occasionally in PGP/MIME (cf. [RFC3156]), the extent of this problem with S/MIME implementations is still unclear. (Note: At this time, none of the samples in [I-D.autocrypt-lamps-protected-headers] apply header protection as specified in Section 3.1 of [RFC8551], which is wrapping as Media Type "message/RFC822".)

Should serious backward compatibility issues with rendering at the receiving side be discovered, the Legacy Display format described in [I-D.autocrypt-lamps-protected-headers] may serve as a basis to mitigate those issues (cf. Section 4.2).

Another variant of backward compatibility has been implemented by pEp [I-D.pep-email], i.e. pEp Email Format 1.0. At this time pEp has implemented this for PGP/MIME, but not yet S/MIME.

5. Security Considerations

[[TODO]]

6. Privacy Considerations

[[TODO]]

7. IANA Considerations

This document requests no action from IANA.

[[RFC Editor: This section may be removed before publication.]]

8. Acknowledgments

The authors would like to thank the following people who have provided helpful comments and suggestions for this document: Berna Alp, Claudio Luck, David Wilson, Hernani Marques, juga, Krista Bennett, Kelly Bristol, Lars Rohwedder, Robert Williams, Russ Housley, Sofia Balicka, Steve Kille, Volker Birk, and Wei Chuang.

9. References

9.1. Normative References

[I-D.dkg-lamps-e2e-mail-guidance]
Gillmor, D., "Guidance on End-to-End E-mail Security", Work in Progress, Internet-Draft, draft-dkg-lamps-e2e-mail-guidance-00, 31 October 2020, <http://www.ietf.org/internet-drafts/draft-dkg-lamps-e2e-mail-guidance-00.txt>.

[I-D.ietf-lamps-header-protection-requirements]
Melnikov, A. and B. Hoeneisen, "Problem Statement and Requirements for Header Protection", Work in Progress, Internet-Draft, draft-ietf-lamps-header-protection-requirements-01, 29 October 2019, <http://www.ietf.org/internet-drafts/draft-ietf-lamps-header-protection-requirements-01.txt>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[RFC2049] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples", RFC 2049, DOI 10.17487/RFC2049, November 1996, <https://www.rfc-editor.org/info/rfc2049>.

[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>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322,

DOI 10.17487/RFC5322, October 2008,
<https://www.rfc-editor.org/info/rfc5322>.

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

9.2. Informative References

[I-D.autocrypt-lamps-protected-headers]
Einarsson, B., juga, j., and D. Gillmor, "Protected
Headers for Cryptographic E-mail", Work in Progress,
Internet-Draft, draft-autocrypt-lamps-protected-headers-
02, 20 December 2019, <http://www.ietf.org/internet-
drafts/draft-autocrypt-lamps-protected-headers-02.txt>.

[I-D.melnikov-iana-reg-forwarded]
Melnikov, A. and B. Hoeneisen, "IANA Registration of
Content-Type Header Field Parameter 'forwarded'", Work in
Progress, Internet-Draft, draft-melnikov-iana-reg-
forwarded-00, 4 November 2019, <http://www.ietf.org/
internet-drafts/draft-melnikov-iana-reg-forwarded-00.txt>.

[I-D.pep-email]
Marques, H., "pretty Easy privacy (pEp): Email Formats and
Protocols", Work in Progress, Internet-Draft, draft-pep-
email-00, 10 July 2020, <http://www.ietf.org/internet-
drafts/draft-pep-email-00.txt>.

[pEp.mixnet]
pEp Foundation, "Mixnet", June 2020,
<https://dev.pep.foundation/Mixnet>.

[RFC3156] Elkins, M., Del Torto, D., Levien, R., and T. Roessler,
"MIME Security with OpenPGP", RFC 3156,
DOI 10.17487/RFC3156, August 2001,
<https://www.rfc-editor.org/info/rfc3156>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2",
FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007,
<https://www.rfc-editor.org/info/rfc4949>.

[RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed.,
"DomainKeys Identified Mail (DKIM) Signatures", STD 76,
RFC 6376, DOI 10.17487/RFC6376, September 2011,
<https://www.rfc-editor.org/info/rfc6376>.

[RFC6409] Gellens, R. and J. Klensin, "Message Submission for Mail", STD 72, RFC 6409, DOI 10.17487/RFC6409, November 2011, <https://www.rfc-editor.org/info/rfc6409>.

[RFC7489] Kucherawy, M., Ed. and E. Zwicky, Ed., "Domain-based Message Authentication, Reporting, and Conformance (DMARC)", RFC 7489, DOI 10.17487/RFC7489, March 2015, <https://www.rfc-editor.org/info/rfc7489>.

Appendix A. Additional information

A.1. Stored Variants of Messages with Bcc

Messages containing at least one recipient address in the Bcc header field may appear in up to three different variants:

1. The Message for the recipient addresses listed in To or Cc header fields, which must not include the Bcc header field neither for signature calculation nor for encryption.

2. The Message(s) sent to the recipient addresses in the Bcc header field, which depends on the implementation:

   a) One Message for each recipient in the Bcc header field separately, with a Bcc header field containing only the address of the recipient it is sent to.

   b) The same Message for each recipient in the Bcc header field with a Bcc header field containing an indication such as "Undisclosed recipients", but no addresses.

   c) The same Message for each recipient in the Bcc header field which does not include a Bcc header field (this Message is identical to 1. / cf. above).

3. The Message stored in the 'Sent'-Folder of the sender, which usually contains the Bcc unchanged from the original Message, i.e., with all recipient addresses.

The most privacy preserving method of the alternatives (2a, 2b, and 2c) is to standardize 2a, as in the other cases (2b and 2c), information about hidden recipients is revealed via keys. In any case, the Message has to be cloned and adjusted depending on the recipient.

Appendix B. Text Moved from Above

Note: Per an explicit request by the chair of the LAMPS WG to only present one option for the specification, the following text has been stripped from the main body of the draft. It is preserved in an Appendix for the time being and may be moved back to the main body or deleted, depending on the decision of the LAMPS WG.

B.1. MIME Format

Currently there are two options in discussion:

1. The option according to the current S/MIME specification (cf. [RFC8551])

2. An alternative option that is based on the former "memory hole" approach (cf. [I-D.autocrypt-lamps-protected-headers])

B.1.1. S/MIME Specification

Note: This is currently described in the main part of this document.

B.1.1.1. Alternative Option Autocrypt "Protected Headers" (Ex-"Memory Hole")

An alternative option (based on the former autocrypt "Memory Hole" approach) to be considered, is described in [I-D.autocrypt-lamps-protected-headers].

Unlike the option described in Appendix B.1.1, this option does not use a "message/RFC822" wrapper to unambiguously delimit the Inner Message.

Before choosing this option, the following two issues must be assessed to ensure no interoperability issues result from it:

1. How current MIME parser implementations treat non-MIME Header Fields, which are not part of the outermost MIME entity and not part of a Message wrapped into a MIME entity of media type "message/rfc822", and how such Messages are rendered to the user.

   [I-D.autocrypt-lamps-protected-headers] provides some examples for testing this.

2. MIME-conformance, i.e. whether or not this option is (fully) MIME-conformant [RFC2045] ff., in particular also Section 5.1. of [RFC2046] on "Multipart Media Type). In the following an excerpt of paragraphs that may be relevant in this context:

> The only header fields that have defined meaning for body parts are those the names of which begin with "Content-". All other header fields may be ignored in body parts. Although they should generally be retained if at all possible, they may be discarded by gateways if necessary. Such other fields are permitted to appear in body parts but must not be depended on. "X-" fields may be created for experimental or private purposes, with the recognition that the information they contain may be lost at some gateways.
>
> NOTE: The distinction between an RFC 822 Message and a body part is subtle, but important. A gateway between Internet and X.400 mail, for example, must be able to tell the difference between a body part that contains an image and a body part that contains an encapsulated Message, the body of which is a JPEG image. In order to represent the latter, the body part must have "Content-Type: message/rfc822", and its body (after the blank line) must be the encapsulated Message, with its own "Content-Type: image/jpeg" header field. The use of similar syntax facilitates the conversion of Messages to body parts, and vice versa, but the distinction between the two must be understood by implementors. (For the special case in which parts actually are Messages, a "digest" subtype is also defined.)

The MIME structure of an Email Message looks as follows:

<Outer Message Header Section (unprotected)>

<Outer Message Body (protected)>

<Inner Message Header Section>

<Inner Message Body>

The following example demonstrates how an Original Message might be protected, i.e., the Original Message is contained as Inner Message in the Protected Body of an Outer Message. It illustrates the first Body part (of the Outer Message) as a "multipart/signed" (application/pkcs7-signature) media type:

Lines are prepended as follows:

* "O: " Outer Message Header Section

* "I: " Message Header Section

```
O: Date: Mon, 25 Sep 2017 17:31:42 +0100 (GMT Daylight Time)
O: Message-ID: <e4a483cb-1dfb-481d-903b-298c92c21f5e@m.example.net>
O: Subject: Meeting at my place
O: From: "Alexey Melnikov" <alexey.melnikov@example.net>
O: MIME-Version: 1.0
O: Content-Type: multipart/signed; charset=us-ascii; micalg=sha1;
O:  protocol="application/pkcs7-signature";
O:  boundary=boundary-AM

   This is a multipart message in MIME format.
   --boundary-AM
I: Date: Mon, 25 Sep 2017 17:31:42 +0100 (GMT Daylight Time)
I: From: "Alexey Melnikov" <alexey.melnikov@example.net>
I: Message-ID: <e4a483cb-1dfb-481d-903b-298c92c21f5e@m.example.net>
I: MIME-Version: 1.0
I: MMHS-Primary-Precedence: 3
I: Subject: Meeting at my place
I: To: somebody@example.net
I: X-Mailer: Isode Harrier Web Server
I: Content-Type: text/plain; charset=us-ascii

   This is an important message that I don't want to be modified.

   --boundary-AM
   Content-Transfer-Encoding: base64
   Content-Type: application/pkcs7-signature

   [[base-64 encoded signature]]

   --boundary-AM--
```

The Outer Message Header Section is unprotected, while the remainder (Outer Message Body) is protected. The Outer Message Body consists of the Inner Message (Header Section and Body).

The Inner Message Header Section is the same as (or a subset of) the Original Message Header Section (cf. Section 4.1.2.1).

The Inner Message Body is the same as the Original Message Body.

The Original Message itself may contain any MIME structure.

Appendix C. Document Changelog

[[RFC Editor: This section is to be removed before publication]]

* draft-ietf-lamps-header-protection-02

- editorial changes / improve language

* draft-ietf-lamps-header-protection-01

  - Add DKG as co-author

  - Partial Rewrite of Abstract and Introduction [HB/AM/DKG]

  - Adding definiations for Cryptographic Layer, Cryptographic Payload, and Cryptographic Envelope (reference to [I-D.dkg-lamps-e2e-mail-guidance]) [DKG]

  - Enhanced MITM Definition to include Machine- / Meddler-in-the-middle [HB]

  - Relaxed definition of Original message, which may not be of type "message/rfc822" [HB]

  - Move "memory hole" option to the Appendix (on request by Chair to only maintain one option in the specification) [HB]

  - Updated Scope of Protection Levels according to WG discussion during IETF-108 [HB]

  - Obfuscation recommendation only for Subject and Message-Id and distinguish between Encrypted and Unencrypted Messages [HB]

  - Removed (commented out) Header Field Flow Figure (it appeared to be confusing as is was) [HB]

* draft-ietf-lamps-header-protection-00

  - Initial version (text partially taken over from [I-D.ietf-lamps-header-protection-requirements]

Appendix D. Open Issues

[[RFC Editor: This section should be empty and is to be removed before publication.]]

* Ensure "protected header" (Ex-Memory-Hole) option is (fully) compliant with the MIME standard, in particular also [RFC2046], Section 5.1. (Multipart Media Type) Appendix B.1.1.1.

* More examples (e.g. in Section 4.1.2.5)

* Should Outer Message Header Section (as received) be preserved for the user? (Section 4.1.3.2.2)

* Decide on whether or not merge requirements from [I-D.ietf-lamps-header-protection-requirements] into this document.

* Decide what parts of [I-D.autocrypt-lamps-protected-headers] to merge into this document.

* Enhance Introduction Section 1 and Problem Statement (Section 2).

* Decide on whether or not specification for more legacy HP requirements should be added to this document (Section 3.1.2).

* Verify simple backward compatibility case (Receiving Side MIME-Conformant) is working; once solution is stable and update paragraphs in Section 4.1, Section 3.1.2.1 and Section 4.2.1 accordingly.

* Verify ability to distinguish between Messages with Header Protection as specified in this document and legacy clients and update Section 3.1 accordingly.

* Improve definitions of Protection Levels and enhance list of Protection Levels (Section 3.2, Section 4).

* Privacy Considerations Section 6

* Security Considerations Section 5

Authors' Addresses

Bernie Hoeneisen
pEp Foundation
Oberer Graben 4
CH- CH-8400 Winterthur
Switzerland

Email: bernie.hoeneisen@pep.foundation
URI: https://pep.foundation/

Daniel Kahn Gillmor
American Civil Liberties Union
125 Broad St.
New York, NY, 10004
United States of America

Email: dkg@fifthhorseman.net

Alexey Melnikov
Isode Ltd
14 Castle Mews
Hampton, Middlesex
TW12 2NP
United Kingdom

Email: alexey.melnikov@isode.com

LAMPS Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

H. Brockhaus, Ed.
S. Fries
D. von Oheimb
Siemens

November 2, 2020

Lightweight CMP Profile
draft-ietf-lamps-lightweight-cmp-profile-04

Abstract

The goal of this document is to facilitate interoperability and automation by profiling the Certificate Management Protocol (CMP) version 2, the related Certificate Request Message Format (CRMF) version 2, and the HTTP Transfer for the Certificate Management Protocol. It specifies a subset of CMP and CRMF focusing on typical use cases relevant for managing certificates of devices in many industrial and IoT scenarios. To limit the overhead of certificate management for more constrained devices only the most crucial types of operations are specified as mandatory. To foster interoperability in more complex scenarios, other types of operations are specified as recommended or optional.

Table of Contents

1. Introduction

[RFC Editor: please delete]:!!! The change history was moved to Appendix B !!!

[RFC Editor: please delete]: The labels 'RFC-CMP-Alg' and 'RFC-CRMF-Alg' in ASN.1 Syntax needs to be replaced with the RFC numbers of CMP Algorithms [I-D.ietf-lamps-cmp-algorithms] and CRMF Algorithm Requirements Update [I-D.housley-lamps-crmf-update-algs], when available.

This document specifies PKI management operations supporting machine-to-machine and IoT use cases. The focus lies on maximum automation and interoperable implementation of all involved PKI entities from end entities (EE) through an optional Local Registration Authority (LRA) and the RA up to the CA. The profile makes use of the concepts and syntax specified in CMP [RFC4210], CRMF [RFC4211], HTTP transfer for CMP [RFC6712], and CMP Updates [I-D.ietf-lamps-cmp-updates]. Especially CMP and CRMF are very feature-rich standards, while only a limited subset of the specified functionality is needed in many environments. Additionally, the standards are not always precise enough on how to interpret and implement the described concepts. Therefore, this document aims at tailoring and specifying in more detail how to use these concepts to implement lightweight automated certificate management.

1.1. Motivation for profiling CMP

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

Though CMP is a very solid and capable protocol it could be used more widely. The most important reason for not more intense application of CMP appears to be that the protocol is offering a large set of features and options but being not always precise enough and leaving room for interpretation. On the one hand, this makes CMP applicable to a very wide range of scenarios, but on the other hand a full implementation of all options is unrealistic because this would take enormous effort.

Moreover, many details of the CMP protocol have been left open or have not been specified in full preciseness. The profiles specified in Appendix D and E of [RFC4210] offer some more detailed PKI management operations. But the specific needs of highly automated scenarios for a machine-to-machine communication are not covered sufficiently.

As also ETSI and UNISIG already put across, profiling is a way of coping with the challenges mentioned above. To profile means to take advantage of the strengths of the given protocol, while explicitly narrowing down the options it provides to exactly those needed for the purpose(s) at hand and eliminating all identified ambiguities. In this way all the general and applicable aspects of the protocol can be taken over and only the peculiarities of the target scenario need to be dealt with specifically.

Doing such a profiling for a new target environment can be a high effort because the range of available options needs to be well understood and the selected options need to be consistent with each other and with the intended usage scenario. Since most industrial PKI management use cases typically have much in common it is worth sharing this effort, which is the aim of this document. Other standardization bodies can then reference the needed PKI management operations from this document and do not need to come up with individual profiles.

1.2. Motivation for a lightweight profile for CMP

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

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

Due to increasing security in operational networks as well as availability requirements, especially on critical infrastructures and systems with a high volume of certificates, a state-of-the-art certificate management must be constantly available and cost-efficient, which calls for high automation and reliability. The NIST Cyber Security Framework [NIST-CSFW] also refers to proper processes for issuance, management, verification, revocation, and audit for authorized devices, users and processes involving identity and credential management. Such PKI operation according to commonly accepted best practices is also required in IEC 62443-3-3 [IEC62443-3-3] for security level 2 and higher.

Further challenges in many industrial systems are network segmentation and asynchronous communication, where PKI operation is often not deployed on-site but in a more protected environment of a data center or trust center. Certificate management must be able to cope with such network architectures. CMP offers the required flexibility and functionality, namely self-contained messages, efficient polling, and support for asynchronous message transfer with end-to-end security.

1.3. Existing CMP profiles

As already stated, CMP contains profiles with mandatory and optional transactions in the Appendixes D and E of [RFC4210]. Those profiles focus on management of human user certificates and do only partly address the specific needs for certificate management automation for unattended machine or application-oriented end entities.

[RFC4210] specifies in Appendix D the following mandatory PKI management operations (all require support of algorithms was updated by CMP Updates [I-D.ietf-lamps-cmp-updates] and CMP Algorithms Appendix A.1 [I-D.ietf-lamps-cmp-algorithms]; all operations may enroll up to two certificates, one for a locally generated and another optional one for a centrally generated key pair; all require use of certConf/pkiConf messages for confirmation):

- Initial registration/certification; an (uninitialized) end entity requests a (first) certificate from a CA using shared secret based message authentication. The content is similar to the PKI management operation specified in Section 4.1.4 of this document.

- Certificate request; an (initialized) end entity requests another certificate from a CA using signature or shared secret based message authentication. The content is similar to the PKI management operation specified in Section 4.1.2 of this document.

- Key update; an (initialized) end entity requests a certificate from a CA (to update the key pair and/or corresponding certificate that it already possesses) using signature or shared secret based message authentication. The content is similar to the PKI management operation specified in Section 4.1.3 of this document.

Due to the two certificates that may be enrolled and the shared secret based authentication, these PKI management operations focus more on the enrollment of human users at a PKI.

[RFC4210] specifies in Appendix E the following optional PKI management operations (all require support of algorithms was updated by CMP Updates [I-D.ietf-lamps-cmp-updates] and CMP Algorithms Appendix A.1 [I-D.ietf-lamps-cmp-algorithms]):

- Root CA key update; a root CA updates its key pair and produces a CA key update announcement message that can be made available (via some transport mechanism) to the relevant end entities. This operation only supports a push and no pull model. The content is similar to the PKI management operation specified in Section 4.4.2 of this document.

- Information request/response; an end entity sends a general message to the PKI requesting details that will be required for later PKI management operations. The content is similar to the PKI management operation specified in Section 4.4.3 of this document.

- Cross-certification request/response (1-way); creation of a single cross-certificate (i.e., not two at once). The requesting CA MAY choose who is responsible for publication of the cross-certificate created by the responding CA through use of the PKIPublicationInfo control.

- In-band initialization using external identity certificate (this PKI management operation may also enroll up to two certificates and requires use of certConf/pkiConf messages for confirmation as specified in Appendix D of [RFC4210]). An (uninitialized) end entity wishes to initialize into the PKI with a CA, CA-1. It uses, for authentication purposes, a pre-existing identity certificate issued by another (external) CA, CA-X. A trust relationship must already have been established between CA-1 and CA-X so that CA-1 can validate the EE identity certificate signed by CA-X. Furthermore, some mechanism must already have been established within the Personal Security Environment (PSE) of the EE that would allow it to authenticate and verify PKIMessages signed by CA-1. The content is similar to the PKI management operation specified in Section 4.1.1 of this document.

Both Appendixes focus on EE to CA/RA PKI management operations and do not address further profiling of RA to CA communication as typically used for full backend automation.

ETSI makes use of CMP [RFC4210] in its Technical Specification 133 310 [ETSI-TS133310] for automatic management of IPSec certificates in UMTS, LTE, and 5G backbone networks. Since 2010 a dedicated CMP profile for initial certificate enrollment and update operations between EE and RA/CA is specified in that document.

UNISIG has included a CMP profile for certificate enrollment in the subset 137 specifying the ETRAM/ECTS on-line key management for train control systems [UNISIG-Subset137] in 2015.

Both standardization bodies use CMP [RFC4210], CRMF [RFC4211], and HTTP transfer for CMP [RFC6712] to add tailored means for automated PKI management operations for unattended machine or application-oriented end entities.

1.4. Compatibility with existing CMP profiles

The profile specified in this document is compatible with CMP [RFC4210] Appendixes D and E (PKI Management Message Profiles), with the following exceptions:

- signature-based protection is the default protection; an initial PKI management operation may also use HMAC,
- certification of a second key pair within the same PKI management operation is not supported,
- proof-of-possession (POPO) with self-signature of the certTemplate according to [RFC4211] section 4.1 clause 3 is the recommended default POPO method (deviations are possible by EEs when requesting central key generation and by (L)RAs when using raVerified),
- confirmation of newly enrolled certificates may be omitted, and
- all PKI management operations consist of request-response message pairs originating at the EE, i.e., announcement messages are omitted.

The profile specified in this document is compatible with the CMP profile for UMTS, LTE, and 5G network domain security and authentication framework [ETSI-TS133310], except that:

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

The profile specified in this document is compatible with the CMP profile for on-line key management in rail networks as specified in UNISIG Subset-137 [UNISIG-Subset137], except that:

- As stated in Section 4.1.1 a CMP message SHALL only consist of one certificate request (CertReqMsg). As UNISIG Subset-137 Table 6 [UNISIG-Subset137] allows to transport more than one certificate request message, this conflicts with this document.
- There is no automatic revocation specified in this document. As UNISIG Subset-137 Section 6.3.2.1.2 [UNISIG-Subset137] request an automatic certificate revocation by the CA in case of TCP disconnection during certificate distribution, this conflicts with this document.

- As of RFC 4210 [RFC4210] the messageTime is required to be Greenwich Mean Time coded as generalizedTime As UNISIG Subset-137 Table 5 [UNISIG-Subset137] explicitly states that the messageTime in required to be 'UTC time', it is not clear if this means a coding as UTCTime or generalizedTime and if other time zones than Greenwich Mean Time shall be allowed. Therefore, UNISIG Subset-137 [UNISIG-Subset137] may conflict with RFC 4210 [RFC4210]. Both time formats are described in RFC 5280 Section 4.1.2.5 [RFC5280].

- This profile requires usage of the same type of protection for all messages of one PKI management operation. This means, in case the request message is MAC protected, also the response, certConf, and pkiConf messages have a MAC-based protection. As UNISIG Subset-137 Table 5 [UNISIG-Subset137] specifies for the first certificate request MAC protection for all messages send by the client and signature protection for all messages send by the server, this conflicts with this document.

- The usage of caPubs is mainly allowed in combination with MAC protected PKI management operations. UNISIG Subset-137 Table 12 [UNISIG-Subset137] requires to use caPubs. When changing to signature protection of the response using a certificate issued under the root CA that is to be transported in the caPubs field, this is not a secure delivery of this root CA certificate.

1.5. Scope of this document

This document specifies requirements on generating PKI management messages on the sender side. It does not specify strictness of verification on the receiving side and how in detail to handle error cases.

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

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

When in Section 3, Section 4, and Section 5 a field of the ASN.1 syntax as defined in RFC 4210 [RFC4210] and RFC 4211 [RFC4211] is not explicitly specified, it SHOULD not be used by the sending entity. The receiving entity MUST NOT require its absence and if present MUST gracefully handle its presence.

1.6. Structure of this document

Section 2 introduces the general PKI architecture and approach to certificate management using CMP that is assumed in this document. Then it enlists the PKI management operations specified in this document and describes them in general words. The list of supported PKI management operations is divided into mandatory, recommended, and optional ones.

Section 3 profiles the CMP message header, protection, and extraCerts section as they are general elements of CMP messages.

Section 4 profiles the exchange of CMP messages between an EE and the first PKI management entities. There are various flavors of certificate enrollment requests optionally with polling, revocation, error handling, and general support PKI management operations.

Section 5 profiles the exchange between PKI management entities. These are in the first place the forwarding of messages coming from or going to an EE. This includes also initiating delayed delivery of messages, which involves polling. Additionally, it specifies PKI management operations where a PKI management entity manages certificates on behalf of an EE or for itself.

Section 6 outlines different mechanisms for CMP message transfer, namely http-based transfer as already specified in [RFC6712], using an additional TLS layer, or offline file-based transport. CoAP [RFC7252] and piggybacking CMP messages on other protocols is out of scope and left for further documents.

1.7. Convention and Terminology

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

Technical terminology is used in conformance with RFC 4210 [RFC4210], RFC 4211 [RFC4211], RFC 5280 [RFC5280], and IEEE 802.1AR [IEEE802.1AR]. 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.

LRA: Local registration authority, an optional RA system component with proximity to the end entities.

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

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

The following terminology is reused from RFC 4210 [RFC4210] and used as follows:

PKI management operation: All CMP messages belonging to one transaction context. The transaction is identified in the transactionID field of the message header.

PKI management entity: All central PKI entities like LRA, RA and CA.

PKI entity: EEs and PKI management entities

2. Architecture and use cases

2.1. Solution architecture

In order to facilitate secure automatic certificate enrollment if the device hosting an EE is equipped with a manufacturer issued certificate during production. Such a manufacturer issued certificate is installed during production to identify the device throughout its lifetime. This manufacturer certificate can be used to protect the initial enrollment of operational certificates after installation of the EE on site in its operational environment. An operational certificate is issued by the owner or operator of the device to identify the device during operation, e.g., within a security protocol like IPSec, TLS, or SSH. In IEEE 802.1AR [IEEE802.1AR] a manufacturer certificate is called IDevID certificate and an operational certificate is called LDevID certificate.

All certificate management transactions specified in this document are initiated by the EE. The EE creates a CMP request message, protects it using some asymmetric or symmetric credential, as far as available, and sends it to its locally reachable PKI component. This PKI component may be an LRA, RA, or the CA, which checks the request, responds to it itself, or forwards the request upstream to the next PKI component. In case an (L)RA changes the CMP request message header or body or wants to prove a successful verification or authorization, it can apply a protection of its own. Especially the communication between an LRA and RA can be performed synchronously or asynchronously. Synchronous communication describes a timely uninterrupted communication between two communication partners, while asynchronous communication is not performed in a timely consistent manner, e.g., because of a delayed message delivery.

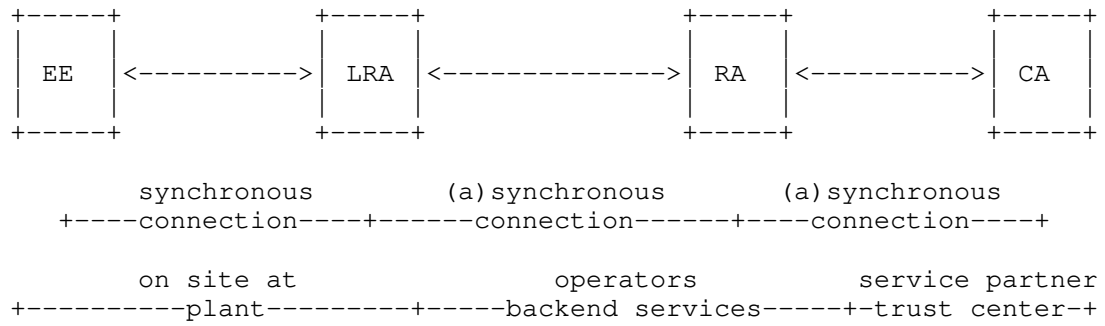

Figure 1: Certificate management on site

In operation environments a layered LRA-RA-CA architecture can be deployed, e.g., with LRAs bundling requests from multiple EEs at dedicated locations and one (or more than one) central RA aggregating the requests from multiple LRAs. Every (L)RA in this scenario typically has a shared key for password-based protection or a CMP signer key and certificate containing an extended key usage as specified in CMP Updates [I-D.ietf-lamps-cmp-updates] allowing it to protect CMP messages it processes. The figure above shows an architecture using one LRA and one RA. It is also possible to have only an RA or multiple LRAs and/or RAs. Depending on the network infrastructure, the communication between different PKI management entities may be synchronous online communication, delayed asynchronous communication, or even offline file transfer.

This profile focusses on specifying the pull model, where the EE always requests a specific PKI management operation.

Note: CMP response messages, especially in case of central key generation, as described in Section 4.1.6, could also be used proactively to implement the push model towards the EE.

Third-party CAs typically implement different variants of CMP or even use proprietary interfaces for certificate management. Therefore, the LRA or the RA may need to adapt the exchanged CMP messages to the flavor of communication required by the CA.

2.2. Basic generic CMP message content

Section 3 specifies the generic parts of the CMP messages as used later in Section 4 and Section 5.

- Header of a CMP message; see Section 3.1.
- Protection of a CMP message; see Section 3.2.
- ExtraCerts field of a CMP message; see Section 3.3.

2.3. Supported PKI management operations

Following the outlined scope from Section 1.5, this section gives a brief overview of the PKI management operations specified in Section 4 and Section 5 and points out whether an implementation by compliant EE or PKI management entities is mandatory, recommended or optional.

2.3.1. Mandatory PKI management operations

The mandatory PKI management operations in this document shall limit the overhead of certificate management. This minimal set of operations may be helpful for keeping development effort low and for use in memory-constrained devices.

PKI management operations	Section
Request a certificate from a new PKI with signature protection	Section 4.1.1
Request to update an existing certificate with signature protection	Section 4.1.3
Error reporting	Section 4.3

Table 1: Mandatory End Entity focused PKI management operations

PKI management operations	Section
Forward messages without changes	Section 5.1.1
Forward messages with replaced protection and keeping the original proof-of-possession	Section 5.1.2.1
Forward messages with replaced protection and raVerified as proof-of-possession	Section 5.1.2.2
Error reporting	Section 5.3

Table 2: Mandatory LRA and RA focused PKI management operations

2.3.2. Recommended PKI management operations

Additional recommended PKI management operations shall support some more complex scenarios, that are considered beneficial for environments with more specific boundary conditions.

PKI management operations	Section
Request a certificate from a PKI with MAC protection	Section 4.1.4
Revoke an own certificate	Section 4.2

Table 3: Recommended End Entity focused PKI management operations

PKI management operations	Section
Revoke another's entities certificate	Section 5.2

Table 4: Recommended LRA and RA focused PKI management operations

2.3.3. Optional PKI management operations

The optional PKI management operations support specific requirements seen only in a subset of environments.

PKI management operations	Section
Request a certificate from a trusted PKI with signature protection	Section 4.1.2
Request a certificate from a legacy PKI using a PKCS#10 [RFC2986] request	Section 4.1.5
Add central generation of a key pair to a certificate request. (If central key generation is supported, the key agreement key management technique is REQUIRED to be supported, and the key transport and password-based key management techniques are OPTIONAL.)	Section 4.1.6
Handle delayed enrollment due to asynchronous message delivery	Section 4.1.7
Additional support messages - distribution of CA certificates, update of a root CA certificate and provisioning of certificate request template	Section 4.4

Table 5: Optional End Entity focused PKI management operations

PKI management operations	Section
Forward messages with additional protection	Section 5.1.3
Initiate delayed enrollment due to asynchronous message delivery	Section 5.1.4

Table 6: Optional LRA and RA focused PKI management operations

2.4. CMP message transport

On different links between PKI entities, e.g., EE<->RA and RA<->CA, different transport MAY be used. As CMP has only very limited requirement regarding the mechanisms used for message transport and in different environments different transport mechanisms are supported, e.g., HTTP, CoAP, or even offline files based, this document requires no specific transport protocol to be supported by all conforming implementations.

HTTP transfer is RECOMMENDED to use for all PKI entities, but there is no transport specified as mandatory to be flexible for devices with special constraints to choose whatever transport is suitable.

Transport	Section
Transfer CMP messages using HTTP	Section 6.1

Table 7: Recommended transport operations

Transport	Section
Transfer CMP messages using HTTPS with certificate-based authentication	Section 6.2
Transfer CMP messages using HTTPS with shared-secret based protection	Section 6.3
Offline CMP message transport	Section 6.4
Transfer CMP messages using CoAP	Section 6.5

Table 8: Optional transport operations

3. Generic parts of the PKI message

To reduce redundancy in the text and to ease implementation, the contents of the header, protection, and extraCerts fields of the CMP messages used in the transactions specified in Section 4 and Section 5 are standardized to the maximum extent possible. Therefore, the generic parts of a CMP message are described centrally in this section.

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

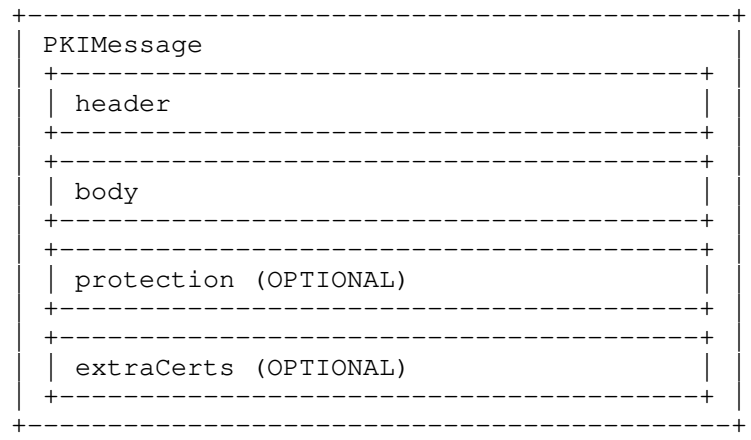

Figure 2: CMP message structure

The general contents of the message header, protection, and extraCerts fields are specified in the Section 3.1 to Section 3.3.

In case a specific CMP message needs different contents in the header, protection, or extraCerts fields, the differences are described in the respective message.

The CMP message body contains the message-specific information. It is described in the context of Section 4 and Section 5.

The behavior in case an error occurs while handling a CMP message is described in Section 5.3.

3.1. General description of the CMP message header

This section describes the generic header field of all CMP messages with signature-based protection. The only variations described here are in the fields recipient, transactionID, and recipNonce of the first message of a PKI management operation.

In case a message has MAC-based protection the changes are described in the respective section. The variations will affect the fields sender, protectionAlg, and senderKID.

For requirements about proper random number generation please refer to [RFC4086]. Any message-specific fields or variations are described in the respective sections of this chapter.

```
   header
     pvno                        REQUIRED
       -- MUST be set to 2 to indicate CMP V2
     sender                      REQUIRED
       -- MUST contain a name representing the originator of the message
       -- SHOULD be the subject of the protection certificate,
       -- the certificate for the private key used to sign the message
     recipient                   REQUIRED
       -- SHOULD be the name of the intended recipient and
       -- MAY be a NULL-DN, i.e., has a zero-length SEQUENCE OF
       -- RelativeDistinguishedNames, if the sender does not know the
       -- DN of the recipient
       -- If this is the first message of a transaction: SHOULD be the
       -- subject of the issuing CA certificate
       -- In all other messages: SHOULD be the same name as in the
       -- sender field of the previous message in this transaction
     messageTime                 RECOMMENDED
       -- MUST be the time at which the message was produced, if
       -- present
     protectionAlg               REQUIRED
       -- MUST be the algorithm identifier of the signature algorithm or
       -- id-PasswordBasedMac algorithm used for calculation of the
       -- protection bits
       -- The signature algorithm MUST be consistent with the
       -- subjectPublicKeyInfo field of the signer's certificate
       -- For more details on cryptographic algorithms to use,
       -- see RFC-CMP-Alg
       algorithm                 REQUIRED
       -- MUST be the OID of the signature algorithm, like
       -- sha256WithRSAEncryption or ecdsa-with-SHA256, or
       -- id-PasswordBasedMac
     senderKID                   RECOMMENDED
       -- MUST be the SubjectKeyIdentifier, if available, of the
       -- protection certificate
     transactionID               REQUIRED
       -- If this is the first message of a transaction:
       -- MUST be 128 bits of random data for the start of a
       -- transaction to reduce the probability of having the
       -- transactionID already in use at the server
       -- In all other messages:
       -- MUST be the value from the previous message in the same
       -- transaction
     senderNonce                 REQUIRED
       -- MUST be cryptographically secure and fresh 128 random bits
     recipNonce                  RECOMMENDED
       -- If this is the first message of a transaction: SHOULD be
       -- absent
       -- In all other messages: MUST be present and contain the value
```

```
      -- from senderNonce of the previous message in the same
      -- transaction
    generalInfo                 OPTIONAL
      implicitConfirm           OPTIONAL
      -- The field is optional though it only applies to
      -- ir/cr/kur/p10cr requests and ip/cp/kup response messages
      -- Add to request messages to request omit sending certConf
      -- message
      -- See [RFC4210] Section 5.1.1.1.
      -- Add to response messages to confirm omit sending certConf
      -- message
        ImplicitConfirmValue    REQUIRED
      -- ImplicitConfirmValue of the request message MUST be NULL if
      -- the EE wants to request not to send a confirmation message
      -- ImplicitConfirmValue MUST be set to NULL if the (L)RA/CA
      -- wants to grant not sending a confirmation message
```

3.2. General description of the CMP message protection

This section describes the generic protection field of all CMP messages with signature-based protection. The certificate for the private key used to sign a CMP message is called 'protection certificate'.

```
protection                      RECOMMENDED
    -- MUST contain the signature calculated using the signature
    -- algorithm specified in protectionAlg
```

Generally, CMP message protection is required for CMP messages, but there are cases where protection of error messages as specified in Section 4.3 and Section 5.3 is not possible and therefore MAY be omitted.

For MAC-based protection as specified in Section 4.1.4 major differences apply as described in the respective section.

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

NOTE: The extended key usages specified in CMP Updates [I-D.ietf-lamps-cmp-updates] can be used for authorization of a sending PKI management entity.

NOTE: The requirements for checking certificates given in [RFC5280] MUST be followed for the CMP message protection. In case the CMP signer certificate is not the CA certificate that signed the newly

issued certificate, certificate status checking SHOULD be used for the CMP signer certificates of communication partners.

3.3. General description of CMP message extraCerts

This section describes the generic extraCerts field of all CMP messages with signature-based protection. If extraCerts are required, recommended, or optional is specified in the respective PKI management operation.

```
extraCerts
    -- SHOULD contain the protection certificate together with its
    -- chain, if needed and the self-signed root certificate SHOULD
    -- be omitted
    -- If present, the first certificate in this field MUST
    -- be the protection certificate and each following certificate
    -- SHOULD directly certify the one immediately preceding it.
    -- Self-signed certificates SHOULD be omitted from extraCerts
    -- and MUST NOT be trusted based on the listing in extraCerts
    -- in any case
```

Note: For maximum compatibility, all implementations SHOULD be prepared to handle potentially additional and arbitrary orderings of the certificates, except that the protection certificate is the first certificate in extraCerts.

4. End Entity focused PKI management operations

This chapter focuses on the communication of the EE and the first PKI management entities it talks to. Depending on the network and PKI solution, this will either be the LRA, the RA or the CA.

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

- Requesting a certificate from a PKI with variations like initial requests and updating, central key generation and different protection means
- Revocation of a certificate
- General messages for further support functions

These operations mainly specify the message body of the CMP messages and utilize the specification of the message header, protection and extraCerts as specified in Section 4.

The behavior in case an error occurs is described in Section 4.3.

This chapter is aligned to Appendix D and Appendix E of [RFC4210]. The general rules for interpretation stated in Appendix D.1 in [RFC4210] need to be applied here, too.

This document does not mandate any specific algorithms implementations must or should support like [ETSI-TS133310] and [UNISIG-Subset137] do. Using the message sequences described here require agreement upon the algorithms to support. A set of algorithms and the respective identifier to use with CMP is available in CMP Algorithms [draft-ietf-lamps-cmp-algorithms].

4.1. Requesting a new certificate from a PKI

There are different approaches to request a certificate from a PKI.

These approaches differ on the one hand in the way the EE can authenticate itself to the PKI it wishes to get a new certificate from and on the other hand in its capabilities to generate a proper new key pair. The authentication means may be as follows:

- Using a certificate from a trusted PKI and the corresponding private key, e.g., a manufacturer issued certificate
- Using the certificate to be updated and the corresponding private key
- Using a shared secret known to the EE and the PKI

Typically, such EE requests a certificate from a CA. When the PKI management entity responds with a message containing a certificate, the EE MUST reply with a confirmation message. The PKI management entity then MUST send confirmation back, closing the transaction.

The message sequences in this section allow the EE to request certification of a locally generated public-private key pair. For requirements about proper random number and key generation please refer to [RFC4086]. The EE MUST provide a signature-based proof-of-possession of the private key associated with the public key contained in the certificate request as defined by [RFC4211] section 4.1 case 3. To this end it is assumed that the private key can technically be used as signing key. The most commonly used algorithms currentlyare RSA and ECDSA, which can technically be used for signature calculation regardless of potentially intended restrictions of the key usage.

The requesting EE provides the binding of the proof-of-possession to its identity by signature-based or MAC-based protection of the CMP request message containing that POPO. The PKI management entity

needs to verify whether this EE is authorized to obtain a certificate with the requested subject and other fields and extensions. Especially when removing the protection provided by the EE and applying a new protection, the PKI management entity MUST verify in particular the included proof-of-possession self-signature of the certTemplate using the public key of the requested certificate and MUST check that the EE, as authenticated by the message protection, is authorized to request a certificate with the subject as specified in the certTemplate (see Section 5.1.2).

There are several ways to install the Root CA certificate of a new PKI on an EE. The installation can be performed in an out-of-band manner, using general messages, a voucher [RFC8366], or other formats for enrollment, or in-band of CMP by the caPubs field in the certificate response message. In case the installation of the new root CA certificate is performed using the caPubs field, the certificate response message MUST be properly authenticated, and the sender of this message MUST be authorized to install new root CA certificates on the EE. This authorization can be indicated by using pre-shared keys for the CMP message protection.

4.1.1. Request a certificate from a new PKI with signature protection

This PKI management operation should be used by an EE to request a certificate of a new PKI using an existing certificate from an external PKI, e.g., a manufacturer issued IDevID certificate [IEEE802.1AR], to prove its identity to the new PKI. The EE already has established trust in this new PKI it is about to enroll to, e.g., by voucher exchange or configuration means. The certificate request message is signature-protected using the existing certificate from the external PKI.

Preconditions:

1 The EE MUST have a certificate enrolled by an external PKI in advance to this PKI management operation to authenticate itself to the PKI management entity using signature-based protection, e.g., using a manufacturer issued certificate.

2 The EE SHOULD know the subject name of the new CA it requests a certificate from; this name MAY be established using an enrollment voucher, the issuer field from a CertReqTemplate response message, or other configuration means. If the EE does not know the name of the CA, the PKI management entity MUST know where to route these requests to.

3 The EE MUST authenticate responses from the PKI management entity; trust MAY be established using an enrollment voucher or other configuration means.

4 The PKI management entity MUST trust the external PKI the EE uses to authenticate itself; trust MAY be established using some configuration means.

The general message flow for this PKI management operation is like that given in [RFC4210] Appendix E.7.

Message flow:

```
Step# EE                                    PKI management entity
  1   format ir
  2                      ->   ir      ->
  3                                        handle, re-protect or
                                             forward ir
  4                                        format or receive ip
  5                                        possibly grant implicit
                                             confirm
  6                      <-   ip      <-
  7   handle ip
  8                                        In case of status
                                             "rejection" in the
                                             ip message, no certConf
                                             and pkiConf are sent
  9   format certConf (optional)
 10                      ->   certConf ->
 11                                        handle, re-protect or
                                             forward certConf
 12                                        format or receive pkiConf
 13                      <-   pkiconf  <-
 14   handle pkiConf (optional)
```

For this PKI management operation, the EE MUST include exactly one single CertReqMsg in the ir. If more certificates are required, further requests MUST be sent using separate CMP messages. If the EE wants to omit sending a certificate confirmation message after receiving the ip to reduce the number of protocol messages exchanged in this PKI management operation, it MUST request this by including the implicitConfirm extension in the ir.

If the CA accepts the certificate request it MUST return the new certificate in the certifiedKeyPair field of the ip message. If the EE requested to omit sending a certConf message after receiving the ip, the PKI management entity MAY confirm it by also including the

implicitConfirm extension or MAY rejects it by omitting the implicitConfirm field in the ip.

If the EE did not request implicit confirmation or the request was not granted by the PKI management entity the confirmation as follows MUST be performed. If the EE successfully receives the certificate and accepts it, the EE MUST send a certConf message, which MUST be answered by the PKI management entity with a pkiConf message. If the PKI management entity does not receive the expected certConf message in time it MUST handle this like a rejection by the EE.

If the certificate request was rejected by the CA, the PKI management entity must return an ip message containing the status code "rejection" and no certifiedKeyPair field. Such an ip message MUST NOT be followed by the certConf and pkiConf messages and the transaction MUST be terminated.

Detailed message description:

Certification Request -- ir

```
Field                          Value

header
    -- As described in section 3.1

body
    -- The request of the EE for a new certificate
  ir                           REQUIRED
    -- MUST be exactly one CertReqMsg
    -- If more certificates are required, further requests MUST be
    -- packaged in separate PKI Messages
    certReq                    REQUIRED
      certReqId                REQUIRED
    -- MUST be set to 0
      certTemplate             REQUIRED
        version                OPTIONAL
    -- MUST be 2 if supplied.
        subject                REQUIRED
    -- The EE subject name MUST be carried in the subject field
    -- and/or the subjectAltName extension.
    -- If subject name is present only in the subjectAltName
    -- extension, then the subject field MUST be a NULL-DN
        publicKey              REQUIRED
          algorithm            REQUIRED
    -- MUST include the subject public key algorithm ID and value
    -- In case a central key generation is requested, this field
    -- contains the algorithm and parameter preferences of the
```

```
      -- requesting entity regarding the to-be-generated key pair
         subjectPublicKey    REQUIRED
      -- MUST contain the public key to be included into the requested
      -- certificate in case of local key-generation
      -- MUST contain a zero-length BIT STRING in case a central key
      -- generation is requested
       extensions            OPTIONAL
      -- MAY include end-entity-specific X.509 extensions of the
      -- requested certificate like subject alternative name,
      -- key usage, and extended key usage
      -- The subjectAltName extension MUST be present if the EE
      -- subject name includes a subject alternative name.
      Popo                   REQUIRED
        POPOSigningKey       OPTIONAL
      -- MUST be used in case subjectPublicKey contains a public key
      -- MUST be absent in case subjectPublicKey contains a
      -- zero-length BIT STRING
       poposkInput           PROHIBITED
      -- MUST NOT be used because subject and publicKey are both
      -- present in the certTemplate
       algorithmIdentifier   REQUIRED
      -- The signature algorithm MUST be consistent with the
      -- publicKey field of the certTemplate
       signature             REQUIRED
      -- MUST be the signature computed over the DER-encoded
      -- certTemplate

   protection                REQUIRED
      -- As described in section 3.2

   extraCerts                REQUIRED
      -- As described in section 3.3


   Certification Response -- ip

   Field                     Value

   header
      -- As described in section 3.1

   body
      -- The response of the CA to the request as appropriate
     ip                      REQUIRED
      caPubs                 OPTIONAL
      -- MAY be used
      -- If used it MUST contain only the root certificate of the
      -- certificate contained in certOrEncCert
```

```
      response                  REQUIRED
      -- MUST be exactly one CertResponse
        certReqId               REQUIRED
      -- MUST be set to 0
        status                  REQUIRED
      -- PKIStatusInfo structure MUST be present
          status                REQUIRED
      -- positive values allowed: "accepted", "grantedWithMods"
      -- negative values allowed: "rejection"
          statusString          OPTIONAL
      -- MAY be any human-readable text for debugging, logging or to
      -- display in a GUI
          failInfo              OPTIONAL
      -- MUST be present if status is "rejection"
      -- MUST be absent if the status is "accepted" or
      -- "grantedWithMods"
        certifiedKeyPair        OPTIONAL
      -- MUST be present if status is "accepted" or "grantedWithMods"
      -- MUST be absent if status is "rejection"
          certOrEncCert         REQUIRED
      -- MUST be present when certifiedKeyPair is present
            certificate         REQUIRED
      -- MUST be present when certifiedKeyPair is present
      -- MUST contain the newly enrolled X.509 certificate
          privateKey            OPTIONAL
      -- MUST be absent in case of local key-generation
      -- MUST contain the encrypted private key in an EnvelopedData
      -- structure as specified in section 5.1.5 in case the private
      -- key was generated centrally

   protection                   REQUIRED
       -- As described in section 3.2

   extraCerts                   REQUIRED
       -- As described in section 3.3
       -- MUST contain the chain of the certificate present in
       -- certOrEncCert, the self-signed root certificate SHOULD be
       -- omitted
       -- Duplicate certificates MAY be omitted


   Certificate Confirmation -- certConf

   Field                        Value

   header
       -- As described in section 3.1
```

```
   body
       -- The message of the EE sends confirmation to the PKI
       -- management entity to accept or reject the issued certificates
     certConf                    REQUIRED
       -- MUST be exactly one CertStatus
       CertStatus                REQUIRED
         certHash                REQUIRED
       -- MUST be the hash of the certificate, using the same hash
       -- algorithm as used to create the certificate signature
         certReqId               REQUIRED
       -- MUST be set to 0
         status                  RECOMMENDED
       -- PKIStatusInfo structure SHOULD be present
       -- Omission indicates acceptance of the indicated certificate
           status                REQUIRED
       -- positive values allowed: "accepted"
       -- negative values allowed: "rejection"
           statusString          OPTIONAL
       -- MAY be any human-readable text for debugging, logging, or to
       -- display in a GUI
           failInfo              OPTIONAL
       -- MUST be present if status is "rejection"
       -- MUST be absent if the status is "accepted"

   protection                    REQUIRED
       -- As described in section 3.2
       -- MUST use the same certificate as for protection of the ir

   extraCerts                    RECOMMENDED
       -- SHOULD contain the protection certificate together with its
       -- chain, but MAY be omitted if the message size is critical and
       -- the PKI management entity did cash the extraCerts from the ir
       -- If present, the first certificate in this field MUST be the
       -- certificate used for signing this message
       -- Self-signed certificates SHOULD NOT be included in
       -- extraCerts and
       -- MUST NOT be trusted based on the listing in extraCerts in
       -- any case


   PKI Confirmation -- pkiconf

   Field                         Value

   header
       -- As described in section 3.1

   body
```

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

   protection                    REQUIRED
       -- As described in section 3.2
       -- SHOULD use the same certificate as for protection of the ip

   extraCerts                    RECOMMENDED
       -- SHOULD contain the protection certificate together with its
       -- chain, but MAY be omitted if the message size is critical and
       -- the PKI management entity did cash the extraCerts from the ip
       -- If present, the first certificate in this field MUST be the
       -- certificate used for signing this message
       -- Self-signed certificates SHOULD NOT be included in extraCerts
       -- and
       -- MUST NOT be trusted based on the listing in extraCerts in
       -- any case
```

4.1.2. Request a certificate from a trusted PKI with signature protection

This PKI management operation should be used by an EE to request an additional certificate of the same PKI it already has certificates from. The EE uses one of these existing certificates to prove its identity. The certificate request message is signature-protected using this certificate.

The general message flow for this PKI management operation is the same as given in Section 4.1.1.

Preconditions:

1 The EE MUST have a certificate enrolled by the PKI it requests another certificate from in advance to this PKI management operation to authenticate itself to the PKI management entity using signature-based protection.

2 The EE SHOULD know the subject name of the CA it requests a certificate from; this name MAY be established using an enrollment voucher, the issuer field from a CertReqTemplate response message, or other configuration means. If the EE does not know the name of the CA, the PKI management entity MUST know where to route this request to.

3 The EE MUST authenticate responses from the PKI management entity; trust MUST be established using an enrollment voucher or other configuration means.

4 The PKI management entity MUST trust the current PKI; trust MAY be established using some configuration means.

The message sequence for this PKI management operation is like that given in [RFC4210] Appendix D.5.

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

1 The body of the first request and response MUST be cr and cp, respectively.

2 The caPubs field in the cp message SHOULD be absent.

4.1.3. Update an existing certificate with signature protection

This PKI management operation should be used by an EE to request an update of one of the certificates it already has and that is still valid. The EE uses the certificate it wishes to update to prove its identity. The certificate request message is signature-protected using this certificate.

The general message flow for this PKI management operation is the same as given in Section 4.1.1.

Preconditions:

1 The certificate the EE wishes to update MUST NOT be expired or revoked.

2 A new public-private key pair SHOULD be used.

The message sequence for this PKI management operation is like that given in [RFC4210] Appendix D.6.

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

1 The body of the first request and response MUST be kur and kup, respectively.

2 Protection of the kur MUST be performed using the certificate to be updated.

3 The subject field and/or the subjectAltName extension of the CertTemplate MUST contain the EE subject name of the existing certificate to be updated, without modifications.

4 The CertTemplate SHOULD contain the subject and publicKey of the EE only.

5 The oldCertId control SHOULD be used to make clear which certificate is to be updated.

6 The caPubs field in the kup message MUST be absent.

As part of the certReq structure of the kur the control is added right after the certTemplate.

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

4.1.4. Request a certificate from a PKI with MAC protection

This PKI management operation should be used by an EE to request a certificate of a new PKI without having a certificate to prove its identity to the target PKI, but there is a shared secret established between the EE and the PKI. Therefore, the initialization request is MAC-protected using this shared secret. The PKI management entity checking the MAC-protection SHOULD replace this protection according to Section 5.1.2 in case the next hop does not know the shared secret.

For requirements with regard to proper random number and key generation please refer to [RFC4086].

The general message flow for this PKI management operation is the same as given in Section 4.1.1.

Preconditions:

1 The EE and the PKI management entity MUST share a symmetric key, this MAY be established by a service technician during initial local configuration.

2 The EE SHOULD know the subject name of the new CA it requests a certificate from; this name MAY be established using an enrollment voucher, the issuer field from a CertReqTemplate response message, or other configuration means. If the EE does not know the name of

the CA, the PKI management entity MUST know where to route this request to.

3 The EE MUST authenticate responses from the PKI management entity; trust MAY be established using the shared symmetric key.

The message sequence for this PKI management operation is like that given in [RFC4210] Appendix D.4.

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

1 The protection of all messages MUST be calculated using Message Authentication Code (MAC); the protectionAlg field MUST be id-PasswordBasedMac as described in section 5.1.3.1 of [RFC4210].

2 The sender MUST contain a name representing the originator of the message. The senderKID MUST contain a reference all participating entities can use to identify the symmetric key used for the protection, e.g., the username of the EE.

3 The extraCerts of the ir, certConf, and pkiConf messages MUST be absent.

4 The extraCerts of the ip message MUST contain the chain of the issued certificate and root certificates SHOULD not be included and MUST NOT be directly trusted in any case.

Part of the protectionAlg structure, where the algorithm identifier MUST be id-PasswordBasedMac, is a PBMParameter sequence. The fields of PBMParameter SHOULD remain constant for message protection throughout this PKI management operation to reduce the computational overhead.

```
      PBMParameter              REQUIRED
        salt                    REQUIRED
      -- MUST be the random value to salt the secret key
        owf                     REQUIRED
      -- MUST be the algorithm identifier for the one-way function
      -- used
      -- For more details on cryptographic algorithms to use, see
      -- RFC-CMP-Alg and RFC-CRMF-Alg
        iterationCount          REQUIRED
      -- MUST be a limited number of times the one-way function is
      -- applied
      -- To prevent brute force and dictionary attacks a reasonable
      -- high number SHOULD be used
        mac                     REQUIRED
      -- MUST be the algorithm identifier of the MAC algorithm used
      -- For more details on cryptographic algorithms to use, see
      -- RFC-CMP-Alg and RFC-CRMF-Alg
```

4.1.5. Request a certificate from a legacy PKI using PKCS#10 request

This PKI management operation should be used by an EE to request a certificate of a legacy PKI only capable to process PKCS#10 [RFC2986] certification requests. The EE can prove its identity to the target PKI by using various protection means as described in Section 4.1.1 or Section 4.1.4.

In contrast to the other PKI management operations described in Section 4.1, this transaction uses PKCS#10 [RFC2986] instead of CRMF [RFC4211] for the certificate request for compatibility reasons with legacy CA systems that require a PKCS#10 certificate request and cannot process CRMF [RFC4211] requests. In such case the PKI management entity MUST extract the PKCS#10 certificate request from the p10cr and provides it separately to the CA.

The general message flow for this PKI management operation is the same as given in Section 4.1.1, but the public key is contained in the subjectPKInfo of the PKCS#10 certificate request.

Preconditions:

1 The EE MUST either have a certificate enrolled from this or any other accepted PKI, or a shared secret known to the PKI and the EE to authenticate itself to the RA.

2 The EE SHOULD know the subject name of the CA it requests a certificate from; this name MAY be established using an enrollment voucher, the issuer field from a CertReqTemplate response message,

or other configuration means. If the EE does not know the name of the CA, the RA MUST know where to route this request to.

3 The EE MUST authenticate responses from the RA; trust MAY be established by an available root certificate, using an enrollment voucher, or other configuration means.

4 The RA MUST trust the current or the PKI the EE uses to authenticate itself; trust MAY be established by a corresponding available root certificate or using some configuration means.

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

1 The body of the first request and response MUST be p10cr and cp, respectively.

2 The certReqId in the cp message MUST be 0.

3 The caPubs field in the cp message SHOULD be absent.

Detailed description of the p10cr message:

Certification Request -- p10cr

```
Field                          Value

header
    -- As described in section 3.1

body
    -- The request of the EE for a new certificate using a PKCS#10
    -- certificate request
  p10cr                        REQUIRED
    certificationRequestInfo   REQUIRED
      version                  REQUIRED
    -- MUST be set to 0 to indicate PKCS#10 V1.7
      subject                  REQUIRED
    -- The EE subject name MUST be carried in the subject field
    -- and/or the subjectAltName extension.
    -- If subject name is present only in the subjectAltName
    -- extension, then the subject field MUST be a NULL-DN
      subjectPKInfo            REQUIRED
        algorithm              REQUIRED
    -- MUST include the subject public key algorithm ID
        subjectPublicKey       REQUIRED
    -- MUST include the subject public key algorithm value
      attributes               OPTIONAL
    -- MAY include end-entity-specific X.509 extensions of the
    -- requested certificate like subject alternative name,
    -- key usage, and extended key usage.
    -- The subjectAltName extension MUST be present if the EE
       -- subject name includes a subject alternative name.
    signatureAlgorithm         REQUIRED
    -- The signature algorithm MUST be consistent with the
    -- subjectPKInfo field.
    signature                  REQUIRED
    -- MUST containing the self-signature for proof-of-possession

protection                     REQUIRED
    -- As described in section 3.2

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

4.1.6. Generate the key pair centrally at the PKI management entity

This functional extension can be applied in combination with certificate enrollment as described in Section 4.1.1, Section 4.1.2, and Section 4.1.4. The functional extension can be used in case an EE is not able or is not willing to generate its new public-private key pair itself. It is a matter of the local implementation which PKI management entity will act as Key Generation Authority (KGA) and perform the key generation. This PKI management entity MUST have a certificate containing the additional extended key usage extension id-kp-cmKGA to be identified by the EE as a legitimate key-generation authority. In case the KGA generated the new key pair on behalf of the EE, it can use Section 4.1.1, Section 4.1.2, or Section 4.1.4 to request the certificate for this key pair as usual.

Generally speaking, in a machine-to-machine scenario it is strongly preferable to generate public-private key pairs locally at the EE. Together with proof-of-possession of the private key in the certification request, this is to make sure that only the entity identified in the newly issued certificate is the only entity who ever holt the private key.

There are some cases where an EE is not able or not willing to locally generate the new key pair. Reasons for this may be the following:

- Lack of sufficient initial entropy.

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

- Due to lack of computational resources, e.g., in case of RSA keys.

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

Note: As mentioned in Section 2.1 central key generation may be required in a push model, where the certificate response message is transferred by the PKI management entity to the EE without receiving a previous request message.

If the EE wishes to request central key generation, it MUST fill the subjectPublicKey field in the certTemplate structure of the request

message with a zero-length BIT STRING. This indicates to the PKI management entity that a new key pair shall be generated centrally on behalf of the EE.

Note: As the protection of centrally generated keys in the response message is being extended from EncryptedValue to EncryptedKey by CMP Updates [I-D.ietf-lamps-cmp-updates], also the alternative EnvelopedData can be used. In CRMF Section 2.1.9 [RFC4211] the use of EncryptedValue has been deprecated in favor of the EnvelopedData structure. Therefore, this profile specifies using EnvelopedData as specified in CMS Section 6 [RFC5652].

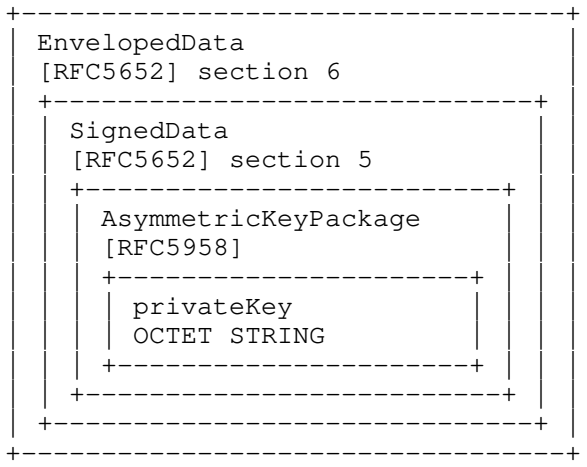

Figure 3: Encrypted private key container

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

The private key MUST be provided as an AsymmetricKeyPackage structure as defined in RFC 5958 [RFC5958].

This AsymmetricKeyPackage structure MUST be wrapped in a SignedData structure, as specified in CMS Section 5 [RFC5652], signed by the KGA generating the key pair. The signature MUST be performed using a CMP signer certificate asserting the extended key usage kp-id-cmKGA as described in CMP Updates [I-D.ietf-lamps-cmp-updates] to show the authorization to generate key pairs on behalf of an EE.

Note: In case of using password-based key management technique as described in Section 4.1.6.3 it may not be possible or meaningful to the EE to validate the KGA signature in the SignedData structure as

shares secrets are used for initial authentication. In this case the EE MAY omit this signature validation.

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

Note: Instead of the specification in CMP Appendix D 4.4 [RFC4210] this content-encryption key is not generated on the EE side. As we just mentioned, central key generation should only be used in this profile in case of lack of randomness on the EE.

As part of the EnvelopedData structure this content-encryption key MUST be securely provided to the EE using one of three key management techniques. The choice of the key management technique to be used by the PKI management entity depends on the authentication mechanism the EE choose to protect the request message, see CMP Updates section 3.4 [I-D.ietf-lamps-cmp-updates] for more details on which key management technique to use.

- Signature protected request message:
  - Using a certificate that contains a key usage extension asserting keyAgreement: The content-encryption key SHALL be protected using the key agreement key management technique, see Section 4.1.6.1, if the certificate used by the EE for signing the respective request message contains the key usage keyAgreement. If the certificate also contains the key usage keyEncipherment, the key transport key management technique SHALL NOT be used.
  - Using a certificate that contains a key usage extension asserting keyEncipherment: The content-encryption key SHALL be protected using the key transport key management technique, see Section 4.1.6.2, if the certificate used by the EE for signing the respective request message contains the key usage keyEncipherment and not keyAgreement.
- MAC protected request message: The content-encryption key SHALL be protected using the password-based key management technique, see Section 4.1.6.3, only if the EE used MAC protection for the respected request message.

The key agreement key management technique can be supported by most signature algorithms, as key transport key management technique can only be supported by a very limited number of algorithms. The password-based key management technique shall only be used in combination with MAC protection, which is a side-line in this

document. Therefore, if central key generation is supported, the support of the key agreement key management technique is REQUIRED and the support of key transport and password-based key management techniques are OPTIONAL.

For details on algorithms to be used, please see CMP Algorithms Section 4 and 5 [I-D.ietf-lamps-cmp-algorithms].

For encrypting the SignedData structure containing the private key a fresh content-encryption key MUST be generated with enough entropy with regard to the used symmetric key-encryption algorithm.

Note: Depending on the lifetime of the certificate and the criticality of the generated private key, it is advisable to use the strongest available symmetric encryption algorithm.

The detailed description of the privateKey field looks like this:

```
    privateKey            OPTIONAL
-- MUST be an EnvelopedData structure as specified in
-- CMS [RFC5652] section 6
   version                REQUIRED
-- MUST be set to 2
   recipientInfos         REQUIRED
-- MUST be exactly one RecipientInfo
     recipientInfo        REQUIRED
-- MUST be either KeyAgreeRecipientInfo (see section 5.1.5.1),
-- KeyTransRecipientInfo (see section 5.1.5.2), or
-- PasswordRecipientInfo (see section 5.1.5.3) is used
-- If central key generation is supported, support of
-- KeyAgreeRecipientInfo is REQUIRED and support of
-- KeyTransRecipientInfo and PasswordRecipientInfo are OPTIONAL
   encryptedContentInfo
                          REQUIRED
     contentType          REQUIRED
-- MUST be id-signedData
     contentEncryptionAlgorithm
                          REQUIRED
-- MUST be the algorithm identifier of the symmetric
-- content-encryption algorithm used
     encryptedContent     REQUIRED
-- MUST be the signedData structure as specified in
-- CMS [RFC5652] section 5 in encrypted form
       version            REQUIRED
-- MUST be set to 3
       digestAlgorithms
                          REQUIRED
-- MUST be exactly one digestAlgorithm identifier
```

```
              digestAlgorithmIdentifier
                              REQUIRED
      -- MUST be the OID of the digest algorithm used for generating
      -- the signature
             encapContentInfo
                              REQUIRED
      -- MUST be the content that is to be signed
              contentType   REQUIRED
      -- MUST be id-ct-KP-aKeyPackage as specified in [RFC5958]
              content       REQUIRED
                AsymmetricKeyPackage
                              REQUIRED
                  OneAsymmetricKey
                              REQUIRED
      -- MUST be exactly one asymmetric key package
                    version REQUIRED
      -- The version MUST be v2
                    privateKeyAlgorithm
                              REQUIRED
      -- The privateKeyAlgorithm field MUST contain
      -- the OID of the asymmetric key pair algorithm
                    privateKey
                              REQUIRED
      -- The privateKey MUST be in the privateKey field
                    Attributes
                              OPTIONAL
      -- The attributes field SHOULD not be used
                    publicKey
                              REQUIRED
      -- The publicKey MUST be in the publicKey field
             certificates    REQUIRED
      -- SHOULD contain the certificate, for the private key used
      -- to sign the content, together with its chain
      -- If present, the first certificate in this field MUST
      -- be the certificate used for signing this content
      -- Self-signed certificates SHOULD NOT be included
      -- and MUST NOT be trusted based on the listing in any case
             crls            OPTIONAL
      -- MAY be present to provide status information on the signer or
      -- its CA certificates
             signerInfos     REQUIRED
      -- MUST be exactly one signerInfo
              version       REQUIRED
      -- MUST be set to 3
              sid           REQUIRED
                subjectKeyIdentifier
                              REQUIRED
      -- MUST be the subjectKeyIdentifier of the signer's certificate
```

```
                digestAlgorithm
                                REQUIRED
        -- MUST be the same OID as in digest AlgorithmIdentifier
                signatureAlgorithm
                                REQUIRED
        -- MUST be the algorithm identifier of the signature algorithm
        -- used for calculation of the signature bits,
        -- like sha256WithRSAEncryption or ecdsa-with-SHA256
        -- The signature algorithm MUST be consistent with the
        -- subjectPublicKeyInfo field of the signer's certificate
                signature     REQUIRED
        -- MUST be the result of the digital signature generation
```

4.1.6.1. Using key agreement key management technique

This key management technique can be applied in combination with the PKI management operations specified in Section 4.1.1 to Section 4.1.3 using signature-based protected CMP messages. The public key of the EE certificate used for the signature-based protection of the request message MUST also be used for the Ephemeral-Static Diffie-Hellmann key establishment of the content-encryption key. To use this key management technique the KeyAgreeRecipientInfo structure MUST be used in the contentInfo field.

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

The detailed description of the KeyAgreeRecipientInfo structure looks like this:

```
          recipientInfo     REQUIRED
    -- MUST be KeyAgreeRecipientInfo as specified in
            version         REQUIRED
    -- MUST be set to 3
            originator      REQUIRED
    -- MUST contain the originatorKey sequence
              algorithm     REQUIRED
    -- MUST be the algorithm identifier of the
    -- static-ephemeral Diffie-Hellmann algorithm
              publicKey     REQUIRED
    -- MUST be the ephemeral public key of the sending party
            ukm             OPTIONAL
    -- MUST be used when 1-pass ECMQV is used
            keyEncryptionAlgorithm
                            REQUIRED
    -- MUST be the same as in the contentEncryptionAlgorithm field
            recipientEncryptedKeys
                            REQUIRED
    -- MUST be exactly one RecipientEncryptedKey element
              recipientEncryptedKey
                            REQUIRED
                rid         REQUIRED
                  rKeyId    REQUIRED
                    subjectKeyID
                            REQUIRED
    -- MUST contain the same value as the senderKID in the
    -- respective request messages
                encryptedKey
                            REQUIRED
    -- MUST be the encrypted content-encryption key
```

4.1.6.2. Using key transport key management technique

This key management technique can be applied in combination with the PKI management operations specified in Section 4.1.1 to Section 4.1.3 using signature-based protected CMP messages. The public key of the EE certificate used for the signature-based protection of the request message MUST also be used for key encipherment of the content-encryption key. To use this key management technique the KeyTransRecipientInfo structure MUST be used in the contentInfo field.

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

The detailed description of the KeyTransRecipientInfo structure looks like this:

```
            recipientInfo     REQUIRED
        -- MUST be KeyTransRecipientInfo as specified in
        -- CMS section 6.2.1 [RFC5652]
                  version         REQUIRED
        -- MUST be set to 2
                  rid             REQUIRED
                    subjectKeyIdentifier
                                  REQUIRED
        -- MUST contain the same value as the senderKID in the respective
        -- request messages
                  keyEncryptionAlgorithm
                                  REQUIRED
        -- MUST contain the key encryption algorithm identifier used for
        -- public key encryption
                  encryptedKey    REQUIRED
        -- MUST be the encrypted content-encryption key
```

4.1.6.3. Using password-based key management technique

This key management technique can be applied in combination with the PKI management operation specified in Section 4.1.4 using MAC protected CMP messages. The shared secret used for the MAC protection MUST also be used for the encryption of the content-encryption key but with a different key derivation algorithm. To use this key management technique the PasswordRecipientInfo structure MUST be used in the contentInfo field.

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

The detailed description of the PasswordRecipientInfo structure looks like this:

```
            recipientInfo     REQUIRED
        -- MUST be PasswordRecipientInfo as specified in
        -- CMS section 6.2.4 [RFC5652]
                  version         REQUIRED
        -- MUST be set to 0
                  keyDerivationAlgorithm
                                  REQUIRED
        -- A key derivation algorithm as specified in RFC-CMP-Alg
        -- SHOULD be used
```

4.1.7. Delayed enrollment

This functional extension can be applied in combination with certificate enrollment as described in Section 4.1.1 to Section 4.1.5. The functional extension can be used in case a PKI management entity cannot respond to the certificate request in a timely manner, e.g., due to offline upstream communication or required registration officer interaction. Depending on the PKI architecture, it is not necessary that the PKI management entity directly communicating with the EE initiates the delayed enrollment.

The PKI management entity initiating the delayed enrollment MUST include the status "waiting" in the response and this response MUST NOT contain a newly issued certificate. When receiving a response with status "waiting" the EE MUST send a poll request to the PKI management entity. The PKI management entity that initiated the delayed enrollment MUST answers with a poll response containing a checkAfter time. This value indicates the minimum number of seconds that must elapse before the EE sends another poll request. As soon as the PKI management entity can provide the final response message for the initial request of the EE, it MUST provide this in response to a poll request. After receiving this response, the EE can continue the original PKI management operation as described in the respective section of this document, e.g., send a certConf message.

Typically, intermediate PKI management entities SHOULD NOT change the sender and recipient nonce even in case an intermediate PKI management entity modifies a request or a response message. In the special case of polling between EE and LRA with offline transport between an LRA and RA, see Section 5.1.4, an exception occurs. The EE and LRA exchange pollReq and pollRep messages handle the nonce words as described. When, after pollRep, the final response from the CA arrives at the LRA, the next response will contain the recipNonce set to the value of the senderNonce in the original request message (copied by the CA). The LRA needs to replace the recipNonce in this case with the senderNonce of the last pollReq because the EE will validate it in this way.

< TBD: I would appreciate any feedback specifically addressing the nonce handling in case an offline LRA responding and not forwarding the pollReq messages. >

Message flow:

```
Step# EE                                      PKI management entity
 1   format ir/cr/p10cr/kur
     As described in the
       respective section
       in this document
 2                    ->ir/cr/p10cr/kur->
 3                                          handle request as described
                                              in the respective section
                                              in this document
 4                                          in case no immediate final
                                              response is possible,
                                              receive or format ip, cp
                                              or kup message containing
                                              status "waiting"
 5                      <-  ip/cp/kup  <-
 6   handle ip/cp/kup
 7   format pollReq
 8                      ->   pollReq   ->
 9                                          handle, re-protect or
                                              forward pollReq
10                                          in case the requested
                                              certificate or a
                                              corresponding response
                                              message is available,
                                              receive or format ip, cp,
                                              or kup containing the
                                              issued certificate, or
                                            format or receive pollRep
                                              with appropriate
                                              checkAfter value
11                      <-   pollRep   <-
12   handle pollRep
13   let checkAfter
       time elapse
14   continue with line 7
```

Detailed description of the first ip/cp/kup:

Response with status 'waiting' -- ip/cp/kup

Field	Value
header	
	-- MUST contain a header as described for the first response
	-- message of the respective PKI management operation

```
   body
       -- The response of the PKI management entity to the request in
       -- case no immediate appropriate response can be sent
     ip/cp/kup                   REQUIRED
       response                  REQUIRED
       -- MUST be exactly one CertResponse
         certReqId               REQUIRED
       -- MUST be set to 0
         status                  REQUIRED
       -- PKIStatusInfo structure MUST be present
           status                REQUIRED
       -- MUST be set to "waiting"
           statusString          OPTIONAL
       -- MAY be any human-readable text for debugging, logging or to
       -- display in a GUI
           failInfo              PROHIBITED
         certifiedKeyPair        PROHIBITED

   protection                    REQUIRED
       -- MUST contain protection as described for the first response
       -- message of the respective PKI management operation, but
       -- MUST use the protection key of the PKI management entity
       -- initiating the delayed enrollment and creating this response
       -- message

   extraCerts                    REQUIRED
       -- MUST contain certificates as described for the first response
       -- message of the respective PKI management operation.
       -- As no new certificate is issued yet, no respective certificate
       -- chain is included


   Polling Request -- pollReq

   Field                         Value

   header
       -- MUST contain a header as described for the certConf message
       -- of the respective PKI management operation

   body
       -- The message of the EE asks for the final response or for a
       -- time to check again
     pollReq                     REQUIRED
       certReqId                 REQUIRED
       -- MUST be exactly one value
       -- MUST be set to 0
```

```
   protection                    REQUIRED
       -- MUST contain protection as described for the certConf message
       -- of the respective PKI management operation

   extraCerts                    OPTIONAL
       -- If present, it MUST contain certificates as described for the
       -- certConf message of the respective PKI management operation


   Polling Response -- pollRep

   Field                         Value

   header
       -- MUST contain a header as described for the pkiConf message
       -- of the respective PKI management operation

   body
       -- The message indicated the time to after which the EE may
       -- send another pollReq messaged for this transaction
     pollRep                     REQUIRED
       -- MUST be exactly one set of the following values
       certReqId                 REQUIRED
       -- MUST be set to 0
       checkAfter                REQUIRED
       -- time in seconds to elapse before a new pollReq may be sent by
       -- the EE

   protection                    REQUIRED
       -- MUST contain protection as described for the pkiConf message
       -- of the respective profile, but
       -- MUST use the protection key of the PKI management entity that
       -- initiated the delayed enrollment and is creating this response
       -- message

   extraCerts                    OPTIONAL
       -- If present, it MUST contain certificates as described for the
       -- pkiConf message of the respective PKI management operation.


   Final response -- ip/cp/kup

   Field                         Value

   header
       -- MUST contain a header as described for the first
       -- response message of the respective PKI management operation,
       -- but the recipNonce MUST be the senderNonce of the last
```

```
        -- pollReq message

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

   protection                    REQUIRED
        -- MUST contain protection as described for the first response
        -- message of the respective PKI management operation, but
        -- MUST use the protection key of the PKI management entity that
        -- initiated the delayed enrollment and forwarding the response
        -- message

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

4.2. Revoking a certificate

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

The revocation request message MUST be signed using the certificate that is to be revoked to prove the authorization to revoke to the PKI. The revocation request message is signature-protected using this certificate.

An EE requests the revocation of an own certificate at the CA that issued this certificate. The PKI management entity responds with a message that contains the status of the revocation from the CA.

Preconditions:

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

Message flow:

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

For this PKI management operation, the EE MUST include exactly one RevDetails structure in the rr message body. In case no error occurred the response to the rr MUST be a rp message. The PKI management entity MUST produce a rp containing a status field with a single set of values.

Detailed message description:

Revocation Request -- rr

```
Field                          Value

header
    -- As described in section 3.1

body
    -- The request of the EE to revoke its certificate
  rr                           REQUIRED
    -- MUST contain exactly one element of type RevDetails
    -- If more revocations are desired, further requests MUST be
    -- packaged in separate PKI Messages
    certDetails                REQUIRED
    -- MUST be present and is of type CertTemplate
      serialNumber             REQUIRED
    -- MUST contain the certificate serialNumber attribute of the
    -- X.509 certificate to be revoked
      issuer                   REQUIRED
    -- MUST contain the issuer attribute of the X.509 certificate to
    -- be revoked
    crlEntryDetails            REQUIRED
    -- MUST contain exactly one reasonCode of type CRLReason (see
    -- [RFC5280] section 5.3.1)
    -- If the reason for this revocation is not known or shall not be
    -- published the reasonCode MUST be 0 = unspecified

protection                     REQUIRED
```

```
       -- As described in section 3.2 and the private key related to the
       -- certificate to be revoked

   extraCerts                    REQUIRED
       -- As described in section 3.3


   Revocation Response -- rp

   Field                         Value

   header
       -- As described in section 3.1

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

   protection                    REQUIRED
       -- As described in section 3.2

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

4.3. Error reporting

This functionality should be used by an EE to report any error conditions upstream to the PKI management entity. Error reporting by a PKI management entity downstream to the EE is described in Section 5.3.

In case the error condition is related to specific details of an ip, cp, or kup response message and a confirmation is expected the error condition MUST be reported in the respective certConf message with negative contents.

General error conditions, e.g., problems with the message header, protection, or extraCerts, and negative feedback on rp, pollRep, or pkiConf messages MAY be reported in the form of an error message.

In both situations the EE reports error in the PKIStatusInfo structure of the respective message.

Depending on the PKI architecture, the PKI management entity MUST forward the error message (upstream) to the next PKI management entity and MUST terminate this PKI management operation.

The PKIStatusInfo structure is used to report errors. The PKIStatusInfo structure SHOULD consist of the following fields:

- status: Here the PKIStatus value "rejection" is the only one allowed.
- statusString: Here any human-readable valid value for logging or to display in a GUI SHOULD be added.
- failInfo: Here the PKIFailureInfo values MAY be used in the following way. For explanation of the reason behind a specific value, please refer to [RFC4210] Appendix F.
  - transactionIdInUse: This is sent by a PKI management entity in case the received request contains a transaction ID that is already in use for another transaction. An EE receiving such error message SHOULD resend the request in a new transaction using a different transaction ID.
  - systemUnavail or systemFailure: This is sent by a PKI management entity in case a back-end system is not available or currently not functioning correctly. An EE receiving such error message SHOULD resend the request in a new transaction after some time.

Detailed error message description:

Error Message -- error

Field	Value
header	
-- As described in section 3.1	
body	
-- The message sent by the EE or the (L)RA/CA to indicate an	
-- error that occurred	
error	REQUIRED
pKIStatusInfo	REQUIRED
status	REQUIRED
-- MUST have the value "rejection"	
statusString	RECOMMENDED
-- SHOULD be any human-readable text for debugging, logging	
-- or to display in a GUI	
failInfo	OPTIONAL
-- MAY be present	
protection	REQUIRED
-- As described in section 3.2	
extraCerts	OPTIONAL
-- As described in section 3.3	

4.4. Support messages

The following support messages offer on demand in-band transport of content that may be provided by the PKI management entity and relevant to the EE. The general messages and general response are used for this purpose. Depending on the environment, these requests may be answered by the LRA, RA, or CA.

The general message and general response transport InfoTypeAndValue structures. In addition to those infoType values defined in CMP [RFC4210] further OIDs MAY be defined to define new PKI management operations, or general-purpose support messages as needed in a specific environment.

Content specified in this document is describs the following:

- Request of CA certificates
- Update of Root CA certificates

o Parameters needed for a planned certificate request message

The PKI management operation is similar to that given in CMP Appendix E.5 [RFC4210]. In this section the general message (genm) and general response (genp) are described. The specific InfoTypeAndValue structures are described in the following sections.

The behavior in case an error occurs is described in Section 4.3.

Message flow:

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

Detailed message description:

General Message -- genm

```
Field                          Value

header
    -- As described in section 3.1

body
    -- The request of the EE to receive information
  genm                         REQUIRED
    -- MUST contain exactly one element of type
    -- InfoTypeAndValue
    infoType                   REQUIRED
    -- MUST be the OID identifying the specific PKI
    -- management operation described below
    infoValue                  OPTIONAL
    -- MUST be as described in the specific PKI
    -- management operation described below

protection                     REQUIRED
    -- As described in section 3.2

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

General Response -- genp

Field	Value
header	
-- As described in section 3.1	
body	
-- The response of the PKI management entity to the	
-- information request	
genp	REQUIRED
-- MUST contain exactly one element of type	
-- InfoTypeAndValue	
infoType	REQUIRED
-- MUST be the OID identifying the specific PKI	
-- management operation described below	
infoValue	OPTIONAL
-- MUST be as described in the specific PKI	
-- management operation described below	
protection	REQUIRED
-- As described in section 3.2	
extraCerts	REQUIRED
-- As described in section 3.3	

< TBD: May be we should not restrict the number of ITAV elements in the response message to one. >

4.4.1. Get CA certificates

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

An EE requests CA certificates from the PKI management entity by sending a general message with OID id-it-caCerts as specified in CMP Updates [I-D.ietf-lamps-cmp-updates]. The PKI management entity responds with a general response with the same OID that either contains a SEQUENCE of certificates populated with the available CA intermediate and issuing CA certificates or with no content in case no CA certificate is available.

The message sequence for this PKI management operation is as given in Section 4.4, with the following specific content:

1 the body MUST contain as infoType the OID id-it-caCerts

2 the infoValue of the request MUST be absent

3 if present, the infoValue of the response MUST be caCerts field

The infoValue field of the general response containing the id-it-caCerts OID looks like this:

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

4.4.2. Get root CA certificate update

This PKI management operation can be used by an EE to request an update of an existing root CA Certificate by the EE.

An EE requests a root CA certificate update from the PKI management entity by sending a general message with OID id-it-rootCaKeyUpdate as specified in CMP Updates [I-D.ietf-lamps-cmp-updates]. The PKI management entity responds with a general response with the same OID that either contains the update of the root CA certificate consisting of up to three certificates, or with no content in case no update is available.

The newWithNew certificate is the new root CA certificates and is REQUIRED to be present in the response message. The newWithOld certificate is RECOMMENDED to be present in the response message though it is needed for those cases where the receiving entity trusts the old root CA certificate and wishes to gain trust in the new root CA certificate. The oldWithNew certificate is OPTIONAL though it is only needed in a scenario where the requesting entity already trusts the new root CA certificate and wants to gain trust in the old root certificate.

The message sequence for this PKI management operation is as given in Section 4.4, with the following specific content:

1 the body MUST contain as infoType the OID id-it-rootCaKeyUpdate

2 the infoValue of the request MUST be absent

3 if present, the infoValue of the response MUST be a RootCaKeyUpdate structure

The infoValue field of the general response containing the id-it-rootCaKeyUpdate extension looks like this:

```
      infoValue               OPTIONAL
    -- MUST be absent if no update of the root CA certificate is
    -- available
    -- MUST be present if an update of the root CA certificate
    -- is available and MUST be of type RootCaKeyUpdate
        newWithNew            REQUIRED
    -- MUST be present if infoValue is present
    -- MUST contain the new root CA certificate
        newWithOld            RECOMMENDED
    -- SHOULD be present if infoValue is present
    -- MUST contain an X.509 certificate containing the new public
    -- root CA key signed with the old private root CA key
        oldWithNew            OPTIONAL
    -- MAY be present if infoValue is present
    -- MUST contain an X.509 certificate containing the old public
    -- root CA key signed with the new private root CA key
```

< TBD: In case the PKI management entity serves for different Root CAs. There are three different options to handle this: - The EE specifies by means of a respective lable in the http endpoint for which Root CA certificate the update is requested. - The EE transfers the oldWithOld certificate in the InfoValue of the request. - The PKI management entity provides RootCaKeyUpdate element all Root CAs an update is available. >

4.4.3. Get certificate request template

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

An EE requests certificate request parameter from the PKI management entity by sending a general message with OID id-it-certReqTemplate as specified in CMP Updates [I-D.ietf-lamps-cmp-updates]. The PKI management entity responds with a general response with the same OID that either contains a certificate template containing requirements on certificate fields and extensions and optionally a sequence of control fields containing requirements on algorithm identifier or RSA key lengths for key pair generation, or with no content in case no specific requirements are made by the PKI.

The EE SHOULD follow the requirements from the received CertTemplate and the optional control fields, by filling in all the fields requested and taking over all the field values provided. The EE SHOULD NOT add further CertTemplate fields, Name components, and extensions or their (sub-)components.

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

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

The EE MUST ignore (i.e., not include and fill in) empty fields, extensions, and sub-components that it does not understand or does not know suitable values to be filled in.

The publicKey field of type SubjectPublicKeyInfo in the CertTemplate MUST NOT be used and MUST contain no algorithm ID in the algorithm field and a zero-length BIT STRING in the subjectPublicKey field. In case the PKI management entity wishes to make stipulation on supported algorithms the EE may use for key generation, this MUST be specified using the control fields.

The control with the OID id-regCtrl-algId, as specified in CMP Updates [I-D.ietf-lamps-cmp-updates], specifies algorithms other that RSA. The algorithm field in SubjectPublicKeyInfo specifies the type of the public key to request a certificate for. The algorithm field contains the key type OID of the public key. For EC keys the full curve information MUST be specified as described in the respective standard documents. The algorithm field MUST be followed by a zero-length BIT STRING for the subjectPublicKey.

The control with the OID id-regCtrl-rsaKeyLen, as specified in CMP Updates [I-D.ietf-lamps-cmp-updates], specifies RSA keys of the specified key length. In case several control fields are present the EE is free to choose one of the specified algorithms for key pair generation. In case no control field is not present the EE is free to choose the public key type and parameters.

In the certTemplate structure the serialNumber, signingAlg, publicKey, issuerUID, and subjectUID fields MUST be omitted.

The message sequence for this PKI management operation is as given in Section 4.4, with the following specific content:

1 the body MUST contain as infoType the OID id-it-certReqTemplate

2 the infoValue of the request MUST be absent

3 if present, the infoValue of the response MUST be a certTemplate structure and an optional SEQUENCE of AttributeTypeAndValue of type id-regCtrl-algId or id-regCtrl-rsaKeyLen

The infoValue field of the general response containing the id-it-certReqTemplate OID looks like this:

```
      InfoValue               OPTIONAL
    -- MUST be absent if no requirements are available
    -- MUST be present if the PKI management entity has any
    -- requirements on the content of the certificates template
        certTemplate          REQUIRED
    -- MUST be present if infoValue is present
    -- MUST contain the prefilled certTemplate structure elements
    -- The SubjectPublicKeyInfo MUST contain no algorithm ID in the
    -- algorithm field and a zero-length BIT STRING in the
    -- subjectPublicKey field
        controls              OPTIONAL
    -- MUST be absent if no requirements on algorithms are available
    -- MUST be present if the PKI management entity has any
    -- requirements on the algorithms to be used for key generation
    -- MUST contain one AttributeTypeAndValue per supported algorithm
    -- with attribute id-regCtrl-algId or id-regCtrl-rsaKeyLen
```

5. LRA and RA focused PKI management operations

This chapter focuses on the communication among different PKI management entities. Depending on the network and PKI solution design, these will either be an LRA, RA or CA.

Typically, a PKI management entity forwards messages from downstream, but it may also reply to them itself. Besides forwarding of received messages a PKI management entity could also need to revoke certificates of EEs, report errors, or may need to manage its own certificates.

5.1. Forwarding of messages

Each CMP request message (i.e., ir, cr, p10cr, kur, pollReq, or certConf) or error message coming from an EE or the previous (downstream) PKI management entity MUST be sent to the next (upstream) PKI management entity. This PKI management entity MUST forward response messages to the next (downstream) PKI management entity or EE.

The PKI management entity SHOULD verify the protection, the syntax, the required message fields, the message type, and if applicable the authorization and the proof-of-possession of the message. Additional checks or actions MAY be applied depending on the PKI solution requirements and concept. If one of these verification procedures fails, the (L)RA SHOULD respond with a negative response message and SHOULD not forward the message further upstream. General error conditions should be handled as described in Section 4.3 and Section 5.3.

A PKI management entity SHOULD not change the received message if not necessary. The PKI management entity SHOULD only update the message protection if it is technically necessary. Concrete PKI system specifications may define in more detail if and when to do so.

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

Each hop in a chain of PKI management entity has one or more functionalities, e.g., a PKI management entity

- may need to verify the identities of EEs or base authorization decisions for certification request processing on specific knowledge of the local setup, e.g., by consulting an inventory or asset management system,
- may need to add fields to certificate request messages,
- may need to store data from a message in a database for later usage or documentation purposes,
- may provide traversal of a network boundary,
- may need to double-check if the messages transferred back and forth are properly protected and well formed,
- may provide a proof that it has performed all required checks,

- may initiate a delayed enrollment due to offline upstream communication or registration officer interaction,
- may grant the request of an EE to omit sending a confirmation message, or
- can collect messages from different LRAs and forward them to the CA.

Therefore, the decision if a message should be forwarded

- unchanged with the original protection,
- unchanged with a new protection, or
- changed with a new protection

depends on the PKI solution design and the associated security policy (CP/CPS [RFC3647]).

This section specifies the different options a PKI management entity may implement and use.

A PKI management entity MAY update the protection of a message

- if it performs changes to the header or the body of the message,
- if it needs to prove checks or validations performed on the message to one of the next (upstream) PKI components,
- if it needs to protect the message using a key and certificate from a different PKI, or
- if it needs to replace or produce a MAC based-protection.

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

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

5.1.1. Not changing protection

This alternative to forward a message can be used by any PKI management entity to forward an original CMP message without changing the header, body or protection. In any of these cases the PKI management entity acts more like a proxy, e.g., on a network boundary, implementing no specific RA-like security functionality to the PKI.

This alternative to forward a message MUST be used for forwarding kur messages that must not be approved by the respective PKI management entity.

5.1.2. Replacing protection

The following two alternatives to forward a message can be used by any PKI management entity to forward a CMP message with or without changes, but providing its own protection using its CMP signer key to assert approval of this message. In this case the PKI management entity acts as an actual Registration Authority (RA), which implements important security functionality of the PKI.

Before replacing the existing protection by a new protection, the PKI management entity MUST verify the protection provided by the EE or by the previous PKI management entity and approve its content including any own modifications. For certificate requests the PKI management entity MUST verify (except in case of central key generation) the presence and contents of the proof-of-possession self-signature of the certTemplate using the public key of the requested certificate and MUST check that the EE, as authenticated by the message protection, is authorized to request a certificate with the subject as specified in the certTemplate.

In case the received message has been protected by a CA or another PKI management entity, the current PKI management entity MUST verify its protection and approve its content including any own modifications. For request messages the PKI management entity MUST check that the other PKI management entity, as authenticated by the protection of the incoming message, was authorized to issue or forward the request.

These message adaptations MUST NOT be applied to kur request messages as described in Section 4.1.3 since their original protection using the key and certificate to be updated needs to be preserved, unless the regCtrl OldCertId is used to clearly identify the certificate to be updated.

These message adaptations MUST NOT be applied to certificate request messages as described in Section 4.1.6requesting key generation by a Key Generation Authority since their original protection using the key and certificate for signature protection or the shared secret for MAC-protection needs to be preserved up to the Key Generation Authority.

In both cases, kur and central key generation, an additional signature of a PKI management entity to the original certificate request message MUST be provided using nested messages as specified in Section 5.1.3.

5.1.2.1. Keeping proof-of-possession

This alternative to forward a message can be used by any PKI management entity to forward a CMP message with or without modifying the message header or body while preserving any included proof-of-possession.

By replacing the existing protection using its own CMP signer key the PKI management entity provides a proof of verifying and approving of the message as described above.

In case the PKI management entity modifies the certTemplate of an ir or cr message, the message adaptation in Section 5.1.2.2 needs to be applied instead.

5.1.2.2. Breaking proof-of-possession

This alternative to forward a message can be used by any PKI management entity to forward an ir or cr message with modifications of the certTemplate i.e., modification, addition, or removal of fields. Such changes will break the proof-of-possession provided by the EE in the original message.

By replacing the existing using its own CMP signer key the PKI management entity provides a proof of verifying and approving the new message as described above.

In addition to the above the PKI management entity MUST verify in particular the proof-of-possession contained in the original message as described above. If these checks were successfully performed the PKI management entity MUST change the popo to raVerified.

The popo field MUST contain the raVerified choice in the certReq structure of the modified message as follows:

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

5.1.3. Adding Protection

This PKI management operation can be used by a PKI management entity to add another protection to one or several PKI management messages. Applying an additional protection is specifically important when forwarding certificate request messages requesting a key update or a central key generation to preserve the original protection of the EE.

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

As specified in the updated Section 5.1.3.4 of RFC4210 [RFC4210] (see CMP Updates [I-D.ietf-lamps-cmp-updates]) there are different use case for adding another protection by a PKI management entity. Specific procedures are described in more detail in the following sections.

The behavior in case an error occurs is described in Section 4.3.

Message flow:

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

Detailed message description:

Nested Message - nested

Field	Value
header	
-- As described in section 3.1	
body	
-- Container to provide additional protection to original	
-- messages and to bundle request or response messages	
PKIMessages	REQUIRED
-- MUST be a sequence of one or more CMP messages	
protection	REQUIRED
-- As described in section 3.2 using the CMP signer key of	
-- the PKI management entity	
extraCerts	REQUIRED
-- As described in section 3.3	

5.1.3.1. Handling a single PKI management message

A PKI management entity may indicate successful validation and authorization of a PKI management message by adding an additional signature to the original PKI management message.

A PKI management entity SHALL wrap the original PKI management messages in a nested message structure. The additional signature as prove of verification and authorization by the PKI management entity MUST be applies as signature-based message protection of the nested message.

5.1.3.2. Handling a batch of PKI management messages

A PKI management entity MAY bundle any number of PKI management messages for batch processing or to transfer a bulk of PKI management messages via an offline interface using the nested message structure. Nested messages can be used on the upstream interface towards the next PKI management entity and/or on the downstream interface from the PKI management entity towards the EE.

This PKI management operation is typically used on the interface between LRA and RA to bundle several PKI management messages for offline transport. In this case the LRA needs to initiate delayed enrollment as described in Section 5.1.4. If the RA may need different routing information per nested PKI management message a

suitable mechanism may need to be implemented. This mechanism strongly depends on the requirements of the target architecture; therefore, it is out of scope of this document.

An initial nested message is generated locally at the PKI management entity. For the initial nested message, the PKI management entity acts as a protocol end point and therefore a fresh transactionId and a fresh senderNonce MUST be used in the header of the nested message. The recipient field MUST identify the PKI management entity that is expected to unpack the nested message. An initial nested message should contain only request messages, e.g., ir, cr, p10cr, kur, certConf, rr, or genm. While building the initial nested message the PKI management entity SHOULD store the transactionIds and the senderNonces of all bundled messages together with the transactionId of the initial nested message.

Such an initial nested message is sent to the next PKI management entity and SHOULD be answered with a responding nested message. This responding message SHOULD use the transactionId of the initial nested message and return the senderNonce of the initial nested message as recipNonce of the responding nested message. The responding nested message SHOULD bundle one response message (e.g. ip, cp, kup, pkiconf, rp, genp, error) for each request message (i.e., for each transactionId) in the initial nested message. While unbundling the responding nested message it is possible to determine lost and unexpected responses based on the previously stored transactionIds and senderNonces. While forwarding the unbundled responses, odd messages SHOULD be dropped, and lost messages should be replaced by an error message to inform the EE about the failed certificate management operation.

The PKI management entity building the nested message applies a signature-based protection using its CMP-signer key as transport protection. This protection SHALL NOT be regarded as prove of verification or authorization of the bundled PKI management messages.

5.1.4. Initiating delayed enrollment

This functional extension can be used by a PKI management entity to initiate delayed enrollment. In this case a PKI management entity MUST add the status waiting in the response message. The PKI management entity MUST then reply to the pollReq messages as described in Section 4.1.7.

5.2. Revoking certificates on behalf of another's entities

This PKI management operation can be used by a PKI management entity to revoke a certificate of any other entity. This revocation request message MUST be signed by the PKI management entity using its own CMP signer key to prove to the PKI authorization to revoke the certificate on behalf of the EE.

Preconditions:

1 the certificate to be revoked MUST be known to the PKI management entity

2 the PKI management entity MUST have the authorization to revoke the certificates of other entities issued by the corresponding CA

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

1 it is not required that the certificate to be revoked is not yet expired or revoked

2 the PKI management entity acts as EE for this message exchange

3 the rr messages MUST be signed using the CMP signer key of the PKI management entity.

5.3. Error reporting

This functionality should be used by the PKI management entity to report any error conditions downstream to the EE. Potential error reporting by the EE upstream to the PKI management entity is described in Section 4.3.

In case the error condition is related to specific details of an ir, cr, p10cr, or kur request message it MUST be reported in the specific response message, i.e., an ip, cp, or kup with negative contents.

General error conditions, e.g., problems with the message header, protection, or extraCerts, and negative feedback on rr, pollReq, certConf, or error messages MUST be reported in the form of an error message.

In both situations the PKI management entity reports the errors in the PKIStatusInfo structure of the respective message as described in Section 4.3.

An EE receiving any such negative feedback SHOULD log the error appropriately and MUST terminate the current transaction.

6. CMP message transport variants

The CMP messages are designed to be self-contained, such that in principle any transport can be used. HTTP SHOULD be used for online transport while file-based transport MAY be used in case offline transport is required. In case HTTP transport is not desired or possible, CMP messages MAY also be piggybacked on any other reliable transport protocol, e.g., CoAP [RFC7252].

Independently of the means of transport it could happen that messages are lost, or a communication partner does not respond. In order to prevent waiting indefinitely, each CMP client component SHOULD use a configurable per-request timeout, and each CMP server component SHOULD use a configurable per-response timeout in case a further message is to be expected from the client side. In this way a hanging transaction can be closed cleanly with an error and related resources (for instance, any cached extraCerts) can be freed.

When conveying a CMP messages in HTTP or MIME-based transport protocols the internet media type "application/pkixcmp" MUST be set for transport encoding as specified in RFC2510 in Section 5.3 [RFC2510] and RFC6712 in Section 3.4 [RFC7712].

6.1. HTTP transport

This transport mechanism can be used by a PKI entity to transfer CMP messages over HTTP. If HTTP transport is used the specifications as described in [RFC6712] and updated by CMP Updates [I-D.ietf-lamps-cmp-updates] MUST be followed.

PKI management operations SHOULD use the following URI path:

PKI management operation	Path	Details
Enroll client to new PKI (REQUIRED)	/initialization	Section 4.1.1
Enroll client to existing PKI (OPTIONAL)	/certification	Section 4.1.2
Update client certificate (REQUIRED)	/keyupdate	Section 4.1.3
Enroll client using PKCS#10 (OPTIONAL)	/p10	Section 4.1.5
Enroll client using central key generation (OPTIONAL)	/serverkeygen	Section 4.1.6
Revoke client certificate (RECOMMENDED)	/revocation	Section 4.2
Get CA certificates (OPTIONAL)	/getcacert	Section 4.4.1
Get root CA certificate update (OPTIONAL)	/getrootupdate	Section 4.4.2
Get certificate request template (OPTIONAL)	/getcertreqtemplate	Section 4.4.3
Additional protection (OPTIONAL)	/nested	Section 5.1.3

Table 9: HTTP endpoints

Subsequent certConf, error, and pollReq messages are sent to the URI of the respective PKI management operation.

The discovery mechanism as described in CMP Updates [I-D.ietf-lamps-cmp-updates] SHOULD be used to query information on the supported PKI management operations, certificate profiles and CAs.

As it is very likely, that a CA supports different certification profiles or that the RA offers PKI management operations for

different issuing CAs, the discovery can also be used to provide the information about these options. The second example listing contains the supported PKI management operations for three different certificate profiles. The supported CA hierarchy consists of one root CA and two issuing CAs.

Detailed message description:

REQ: GET /.well-known/cmp

RES: Content
</cmp/certprofile1/initialization>;ct=pkixcmp
</cmp/certprofile2/initialization>;ct=pkixcmp
</cmp/certprofile3/initialization>;ct=pkixcmp
</cmp/certprofile1/certification >;ct=pkixcmp
</cmp/certprofile2/certification >;ct=pkixcmp
</cmp/certprofile3/certification >;ct=pkixcmp
</cmp/certprofile1/keyupdate >;ct=pkixcmp
</cmp/certprofile2/keyupdate >;ct=pkixcmp
</cmp/certprofile3/keyupdate >;ct=pkixcmp
</cmp/certprofile1/p10>;ct=pkixcmp
</cmp/certprofile2/p10>;ct=pkixcmp
</cmp/certprofile3/p10>;ct=pkixcmp
</cmp/ca1/revocation>;ct=pkixcmp
</cmp/ca2/revocation>;ct=pkixcmp
</cmp/getcacerts>;ct=pkixcmp
</cmp/rootca1/getrootupdate>;ct=pkixcmp
</cmp/certprofile1/getcertreqtemplate >;ct=pkixcmp
</cmp/certprofile2/getcertreqtemplate >;ct=pkixcmp
</cmp/certprofile3/getcertreqtemplate >;ct=pkixcmp

There are different options in the handling of the naming. The PKI management entity either needs to offer the certprofile or CA labels the EE expects. Alternatively, a mechanism is required to configure this information to the EE beforehand.

6.2. HTTPS transport using certificates

This transport mechanism can be used by a PKI entity to further protect the HTTP transport as described in Section 6.1 using TLS 1.2 [RFC5246] or TLS 1.3 [RFC8446] as described in [RFC2818] with certificate-based authentication. Using this transport mechanism, the CMP transport via HTTPS MUST use TLS server authentication and SHOULD use TLS client authentication.

EE:

- The EE SHOULD use a TLS client certificate as far as available. If no dedicated TLS certificate is available, the EE SHOULD use an already existing certificate identifying the EE (e.g., a manufacturer certificate).

- If no TLS certificate is available at the EE, server-only authenticated TLS SHOULD be used.

- The EE MUST validate the TLS server certificate of its communication partner.

PKI management entity:

- Each PKI management entity SHOULD use a TLS client certificate on its upstream (client) interface.

- Each PKI management entity MUST use a TLS server certificate on its downstream (server) interface.

- Each PKI management entity MUST validate the TLS certificate of its communication partners.

NOTE: The requirements for checking certificates given in [RFC5280], [RFC5246] and [RFC8446] MUST be followed for the TLS layer. Certificate status checking SHOULD be used for the TLS certificates of communication partners.

6.3. HTTPS transport using shared secrets

This transport mechanism can be used by a PKI entity to further protect the HTTP transport as described in Section 6.1 using TLS 1.2 [RFC5246] or TLS 1.3 [RFC8446] as described in [RFC2818] with mutual authentication based on shared secrets as described in [RFC5054].

EE:

- The EE MUST use the shared symmetric key for authentication.

PKI management entity:

- The PKI management entity MUST use the shared symmetric key for authentication.

< TBD: It needs to be clarified which cipher suite shall be recommended as there seems to be no support for TLS-SRP un JavaSE. >

6.4. Offline transport

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

The transport mechanism SHOULD be able to indicate message loss, excessive delay, and possibly other transmission errors. In such cases the PKI entities using this mechanism SHOULD report an error as specified in Section 4.3.

6.4.1. File-based transport

CMP messages MAY be transferred between PKI entities using file-system-based mechanisms, for instance when an off-line end entity or a PKI management entity performs delayed enrollment. Each file MUST contain the ASN.1 DER encoding of one CMP message only. There MUST be no extraneous header or trailer information in the file. The file type extensions ".PKI" SHOULD be used.

6.4.2. Other asynchronous transport protocols

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

The ASN.1 DER encoding of the CMP messages MUST be transferred using the "application/pkixcmp" content type and base64-encoded content-transfer-encoding as specified in [RFC2510], section 5.3. A filename MUST be included either in a content-type or a content-disposition statement. The extension for the file MUST be ".PKI".

6.5. CoAP transport

In constrained environments where no HTTP transport is desired or possible, CoAP [RFC7252] as specified in [I-D.msahni-tbd-cmpv2-coap-transport] MAY be used instead.

6.6. Piggybacking on other reliable transport

For online transfer where no HTTP transport is desired or possible CMP messages MAY also be transported on some other reliable protocol. Connection and error handling mechanisms like those specified for HTTP in [RFC6712] need to be implemented.

Such specification is out of scope of this document and would need to be specifies in a separate document, e.g., in the scope of the respective transport protocol used.

7. IANA Considerations

8. Security Considerations

< TBD: Add any security considerations >

9. Acknowledgements

We would like to thank the various reviewers of this document.

10. References

10.1. Normative References

[I-D.housley-lamps-crmf-update-algs]
Housley, R., "Algorithm Requirements Update to the Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", draft-housley-lamps-crmf-update-algs-01 (work in progress), October 2020.

[I-D.ietf-lamps-cmp-algorithms]
Brockhaus, H., "CMP Algorithms", draft-ietf-lamps-cmp-algorithms-00 (work in progress), October 2020.

[I-D.ietf-lamps-cmp-updates]
Brockhaus, H., "CMP Updates", draft-ietf-lamps-cmp-updates-05 (work in progress), September 2020.

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

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

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

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

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

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

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

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

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

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

10.2. Informative References

[ETSI-TS133310]
ETSI, "TS 133 310; Network Domain Security (NDS); Authentication Framework (AF); Release 16; V16.4.0", August 2020, <https://www.etsi.org/deliver/etsi_ts/133300_133399/133310/>.

[I-D.msahni-tbd-cmpv2-coap-transport]
Sahni, M., "CoAP Transport for CMPV2", draft-msahni-tbd-cmpv2-coap-transport-00 (work in progress), June 2020.

[IEC62443-3-3]
IEC, "Industrial communication networks - Network and system security - Part 3-3: System security requirements and security levels", IEC 62443-3-3, August 2013, <https://webstore.iec.ch/publication/7033>.

[IEEE802.1AR]
IEEE, "802.1AR Secure Device Identifier", June 2018, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

[NIST-CSFW]
NIST, "Framework for Improving Critical Infrastructure Cybersecurity Version 1.1", April 2018, <https://www.nist.gov/publications/framework-improving-critical-infrastructure-cybersecurity-version-11>.

[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>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

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

[RFC5054] Taylor, D., Wu, T., Mavrogiannopoulos, N., and T. Perrin, "Using the Secure Remote Password (SRP) Protocol for TLS Authentication", RFC 5054, DOI 10.17487/RFC5054, November 2007, <https://www.rfc-editor.org/info/rfc5054>.

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

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

[RFC7712] Saint-Andre, P., Miller, M., and P. Hancke, "Domain Name Associations (DNA) in the Extensible Messaging and Presence Protocol (XMPP)", RFC 7712, DOI 10.17487/RFC7712, November 2015, <https://www.rfc-editor.org/info/rfc7712>.

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

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

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

[UNISIG-Subset137] UNISIG, "Subset-137; ERTMS/ETCS On-line Key Management FFFIS; V1.0.0", December 2015, <https://www.era.europa.eu/filebrowser/download/542_en>.

Appendix A. Example for CertReqTemplate

< TBD: This Appendix must be updated to reflect the change from using rsaKeyLen to controles. >

This Section provides a concrete example for the content of an infoValue used of type id-it-certReqTemplate as described in Section 4.4.3.

Suppose the server requires that the certTemplate contains the issuer field with a value to be filled in by the EE, the subject field with a common name to be filled in by the EE and two organizational unit fields with given values "myDept" and "myGroup", the publicKey field with an RSA public key of length 2048, the subjectAltName extension with DNS name "www.myServer.com" and an IP address to be filled in, the keyUsage extension marked critical with the value digitalSignature and keyAgreement, and the extKeyUsage extension with values to be filled in by the EE. Then the infoValue with certTemplate and rsaKeyLen returned to the EE must be encoded as follows:

```
SEQUENCE {
  SEQUENCE {
    [3] {
```

```
        SEQUENCE {}
        }
      [5] {
        SEQUENCE {
          SET {
            SEQUENCE {
              OBJECT IDENTIFIER commonName (2 5 4 3)
              UTF8String ''
              }
            }
            SEQUENCE {
              OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
              UTF8String 'myDept'
              }
            }
          SET {
            SEQUENCE {
              OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
              UTF8String 'myGroup'
              }
            }
          }
        }
      [6] {
        SEQUENCE {
          OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
          NULL
          }
        BIT STRING, encapsulates {
          SEQUENCE {}
          }
        }
      [9] {
        SEQUENCE {
          OBJECT IDENTIFIER subjectAltName (2 5 29 17)
          OCTET STRING, encapsulates {
            SEQUENCE {
              [2] 'www.myServer.com'
              [7] ''
              }
            }
          }
        SEQUENCE {
          OBJECT IDENTIFIER keyUsage (2 5 29 15)
          BOOLEAN TRUE
          OCTET STRING, encapsulates {
            BIT STRING 3 unused bits
              '10001'B
```

```
            }
          }
        SEQUENCE {
          OBJECT IDENTIFIER extKeyUsage (2 5 29 37)
          OCTET STRING, encapsulates {
            SEQUENCE {}
            }
          }
        }
      }
    INTEGER 2048
    }
```

Appendix B. History of changes

Note: This appendix will be deleted in the final version of the document.

From version 03 -> 04:

- Deleted normative text sections on algorithms and refer to CMP Algorithms and CRMF Algorithm Requirements Update instead
- Some clarifications and changes in wording

From version 02 -> 03:

- Updated the interoperability with [UNISIG-Subset137] in Section 1.4.
- Changed Section 2.3 to a tabular layout to enhanced readability
- Added a ToDo to section 3.1 on aligning with the CMP Algorithms draft that will be set up as decided in IETF 108
- Updated section 4.1.6 to add the AsymmetricKey Package structure to transport a newly generated private key as decided in IETF 108
- Added a ToDo to section 4.1.7 on required review of the nonce handling in case an offline LRA responds and not forwards the pollReq messages
- Updated Section 4 due to the definition of the new ITAV OIDs in CMP Updates
- Updated Section 4.4.4 to utilize controls instead of rsaKeyLen (see thread "dtaft-ietf-lamps-cmp-updates and rsaKeyLen")

- Deleted the section on definition and discovery of HTTP URIs and copied the text to the HTTP transport section and to CMP Updates section 3.2

- Added some explanation to Section 5.1.2 and Section 5.1.3 on using nested messages when a protection by the RA is required.

- Deleted the section on HTTP URI definition and discovery as some content was moved to CMP Updates. The rest of the content was moved back to the HTTP transport section

- Deleted the ASN.1 module after moving the new OIDs id-it-caCerts, id-it-rootCaKeyUpdate, and id-it-certReqTemplate to CMP Updates

- Minor changes in wording and addition of some open ToDos

From version 01 -> 02:

- Extend Section 1.4 with regard to conflicts with UNISIG Subset-137.

- Minor clarifications on extraCerts in Section 3.3 and Section 4.1.1.

- Complete specification of requesting a certificate from a trusted PKI with signature protection in Section 4.1.2.

- Changed from symmetric key-encryption to password-based key management technique in section Section 4.1.6.3 as discussed on the mailing list (see thread "draft-ietf-lamps-lightweight-cmp-profile-01, section 5.1.6.1")

- Changed delayed enrollment described in Section 4.1.7 from recommended to optional as decided at IETF 107

- Introduced the new RootCAKeyUpdate structure for root CA certificate update in Section 4.4.2 as decided at IETF 107 (also see email thread "draft-ietf-lamps-lightweight-cmp-profile-01, section 5.4.3")

- Extend the description of the CertReqTemplate PKI management operation, including an example added in the Appendix. Keep rsaKeyLen as a single integer value in Section 4.4.3 as discussed on the mailing list (see thread "draft-ietf-lamps-lightweight-cmp-profile-01, section 5.4.4")

- Deleted Sections "Get certificate management configuration" and "Get enrollment voucher" as decided at IETF 107

- Complete specification of adding an additional protection by an PKI management entity in Section 5.1.3.

- Added a section on HTTP URI definition and discovery and extended Section 6.1 on definition and discovery of supported HTTP URIs and content types, add a path for nested messages as specified in Section 5.1.3 and delete the paths for /getCertMgtConfig and /getVoucher

- Changed Section 6.4 to address offline transport and added more detailed specification file-based transport of CMP

- Added a reference to the new I-D of Mohit Sahni on "CoAP Transport for CMPV2" in Section 6.5; thanks to Mohit supporting the effort to ease utilization of CMP

- Moved the change history to the Appendix

- Minor changes in wording

From version 00 -> 01:

- Harmonize terminology with CMP [RFC4210], e.g.,

  * transaction, message sequence, exchange, use case -> PKI management operation

  * PKI component, (L)RA/CA -> PKI management entity

- Minor changes in wording

From draft-brockhaus-lamps-lightweight-cmp-profile-03 -> draft-ietf-lamps-lightweight-cmp-profile-00:

- Changes required to reflect WG adoption

- Minor changes in wording

From version 02 -> 03:

- Added a short summary of [RFC4210] Appendix D and E in Section 1.3.

- Clarified some references to different sections and added some clarification in response to feedback from Michael Richardson and Tomas Gustavsson.

- Added an additional label to the operational path to address multiple CAs or certificate profiles in Section 6.1.

From version 01 -> 02:

- Added some clarification on the key management techniques for protection of centrally generated keys in Section 4.1.6.
- Added some clarifications on the certificates for root CA certificate update in Section 4.4.2.
- Added a section to specify the usage of nested messages for RAs to add an additional protection for further discussion, see Section 5.1.3.
- Added a table containing endpoints for HTTP transport in Section 6.1 to simplify addressing PKI management entities.
- Added some ToDos resulting from discussion with Tomas Gustavsson.
- Minor clarifications and changes in wording.

From version 00 -> 01:

- Added a section to specify the enrollment with an already trusted PKI for further discussion, see Section 4.1.2.
- Complete specification of requesting a certificate from a legacy PKI using a PKCS#10 [RFC2986] request in Section 4.1.5.
- Complete specification of adding central generation of a key pair on behalf of an end entity in Section 4.1.6.
- Complete specification of handling delayed enrollment due to asynchronous message delivery in Section 4.1.7.
- Complete specification of additional support messages, e.g., to update a Root CA certificate or to request an RFC 8366 [RFC8366] voucher, in Section 4.4.
- Minor changes in wording.

From draft-brockhaus-lamps-industrial-cmp-profile-00 -> draft-brockhaus-lamps-lightweight-cmp-profile-00:

- Change focus from industrial to more multi-purpose use cases and lightweight CMP profile.

- Incorporate the omitted confirmation into the header specified in Section 3.1 and described in the standard enrollment use case in Section 4.1.1 due to discussion with Tomas Gustavsson.

- Change from OPTIONAL to RECOMMENDED for use case 'Revoke another's entities certificate' in Section 5.2, because it is regarded as important functionality in many environments to enable the management station to revoke EE certificates.

- Complete the specification of the revocation message flow in Section 4.2 and Section 5.2.

- The CoAP based transport mechanism and piggybacking of CMP messages on top of other reliable transport protocols is out of scope of this document and would need to be specified in another document.

- Further minor changes in wording.

Authors' Addresses

Hendrik Brockhaus (editor)
Siemens AG

Email: hendrik.brockhaus@siemens.com

Steffen Fries
Siemens AG

Email: steffen.fries@siemens.com

David von Oheimb
Siemens AG

Email: david.von.oheimb@siemens.com

LAMPS
Internet-Draft
Updates: 6960 (if approved)
Intended status: Standards Track
Expires: March 14, 2021

M. Sahni, Ed.
Palo Alto Networks

September 10, 2020

OCSP Nonce Extension
draft-ietf-lamps-ocsp-nonce-05

Abstract

This document specifies the updated format of the Nonce extension in the Online Certificate Status Protocol (OCSP) request and response messages. OCSP is used to check the status of a certificate and the Nonce extension is used to cryptographically bind an OCSP response message to a particular OCSP request message. This document updates RFC 6960.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 14, 2021.

Table of Contents

1. Introduction

This document updates the usage and format of the Nonce extension used in OCSP request and response messages. This extension was previously defined in section 4.4.1 of [RFC6960]. [RFC6960] does not mention any minimum and maximum length of nonce in the Nonce extension. Lacking limits on the length of nonce in the Nonce extension, an OCSP responders that follow [RFC6960] may be vulnerable to various attacks like Denial of Service attacks [RFC4732], chosen prefix attacks to get a desired signature, and possible evasions using the Nonce extension data. This document specifies a lower limit of 1 and an upper limit of 32 to the length of nonce in the Nonce extension. This document updates [RFC6960].

1.1. 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.

2. OCSP Extensions

The message format for OCSP request and response is defined in [RFC6960]. [RFC6960] also defines the standard extensions for OCSP messages based on the extension model employed in X.509 version 3

certificates (see [RFC5280]). This document only specifies the new format for Nonce extension and does not change specification of any of the other standard extensions defined in [RFC6960].

2.1. Nonce Extension

This section replaces the entirety of the Section 4.4.1 of [RFC6960] which describes the OCSP Nonce extension.

The nonce cryptographically binds a request and a response to prevent replay attacks. The nonce is included as one of the requestExtensions in requests, while in responses it would be included as one of the responseExtensions. In both the request and the response, the nonce will be identified by the object identifier id-pkix-ocsp-nonce, while the extnValue is the value of the nonce. If Nonce extension is present then the length of nonce MUST be at least 1 octet and can be up to 32 octets.

A server MUST reject any OCSP request having a nonce in the Nonce extension with length of 0 octets or more than 32 octets with the malformedRequest OCSPResponseStatus as described in section 4.2.1 of [RFC6960].

The value of the nonce MUST be generated using a cryptographically strong pseudorandom number generator (see [RFC4086]). The minimum nonce length of 1 octet is defined to provide backward compatibility with older clients that follow [RFC6960]. Newer OCSP clients that support this document MUST use a length of 32 octets for the nonce in Nonce extension. OCSP responders MUST accept lengths of at least 16 octets, and MAY choose to ignore the Nonce extension for requests where the length of the nonce is less than 16 octets

```
id-pkix-ocsp           OBJECT IDENTIFIER ::= { id-ad-ocsp }
id-pkix-ocsp-nonce     OBJECT IDENTIFIER ::= { id-pkix-ocsp 2 }

Nonce ::= OCTET STRING(SIZE(1..32))
```

3. Security Considerations

The security considerations of OCSP, in general, are described in [RFC6960]. During the interval in which the previous OCSP response for a certificate is not expired but the responder has a changed status for that certificate, a copy of that OCSP response can be used to indicate that the status of the certificate is still valid. Including client’s Nonce value in the OCSP response makes sure that the response is the latest response from the server and not an old copy.

3.1. Replay Attack

The Nonce extension is used to avoid replay attacks. Since the OCSP responder may choose to not send the Nonce extension in the OCSP response even if the client has sent the Nonce extension in the request [RFC5019], an on-path attacker can intercept the OCSP request and respond with an earlier response from the server without the Nonce extension. This can be mitigated by configuring the server to use a short time interval between the thisUpdate and nextUpdate fields in the OCSP response.

3.2. Nonce Collision

If the value of nonce used by a client in OCSP request is predictable, then an attacker may prefetch responses with the predicted nonce and can replay them, thus defeating the purpose of using nonce. Therefore the value of Nonce extension in the OCSP request MUST contain cryptographically strong randomness and MUST be freshly generated at the time of creating the OCSP request. Also if the length of nonce is too small e.g. 1 octet then an on-path attacker can prefetch responses with all the possible values of nonce and replay a matching nonce.

4. IANA Considerations

This document does not call for any IANA actions.

5. Changes to Appendix B. of RFC 6960

This section updates the ASN.1 definitions of the OCSP Nonce extension in Appendix B.1 and Appendix B.2 of [RFC6960] The Appendix B.1 defines OCSP using ASN.1 - 1998 Syntax and Appendix B.2 defines OCSP using ASN.1 - 2008 Syntax

5.1. Changes to Appendix B.1. OCSP in ASN.1 - 1998 Syntax

OLD Syntax:

The definition of OCSP Nonce Extension is not provided in Appendix B.1 of [RFC6960] for the ASN.1 - 1998 Syntax.

NEW Syntax:

```
Nonce ::= OCTET STRING(SIZE(1..32))
```

5.2. Changes to Appendix B.2 OCSP in ASN.1 - 2008 Syntax

OLD Syntax:

```
re-ocsp-nonce EXTENSION ::= { SYNTAX OCTET STRING IDENTIFIED
    BY id-pkix-ocsp-nonce }
```

NEW Syntax:

```
re-ocsp-nonce EXTENSION ::= { SYNTAX OCTET STRING(SIZE(1..32))
    IDENTIFIED BY id-pkix-ocsp-nonce }
```

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, <http://www.rfc-editor.org/info/rfc2119>.

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

[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>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/info/rfc6960>.

6.2. Informative References

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, DOI 10.17487/RFC4732, December 2006, <https://www.rfc-editor.org/info/rfc4732>.

[RFC5019] Deacon, A. and R. Hurst, "The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments", RFC 5019, DOI 10.17487/RFC5019, September 2007, <https://www.rfc-editor.org/info/rfc5019>.

Author's Address

Mohit Sahni (editor)
Palo Alto Networks
3000 Tannery Way
Santa Clara, CA 95054
US

Email: msahni@paloaltonetworks.com

LAMPS Working Group
Internet-Draft
Updates: 7030 (if approved)
Intended status: Standards Track
Expires: February 12, 2021

M. Richardson
Sandelman Software Works
T. Werner
Siemens
W. Pan
Huawei Technologies

August 11, 2020

Clarification of Enrollment over Secure Transport (EST): transfer encodings and ASN.1
draft-ietf-lamps-rfc7030est-clarify-10

Abstract

This document updates RFC7030: Enrollment over Secure Transport (EST) to resolve some errata that were reported, and which have proven to cause interoperability issues when RFC7030 was extended.

This document deprecates the specification of "Content-Transfer-Encoding" headers for EST endpoints. This document fixes some syntactical errors in ASN.1 that were present.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 12, 2021.

Table of Contents

1. Introduction

Enrollment over Secure Transport (EST) is defined in [RFC7030]. The EST specification defines a number of HTTP end points for certificate enrollment and management. The details of the transaction were defined in terms of MIME headers as defined in [RFC2045], rather than in terms of the HTTP protocol as defined in [RFC7230] and [RFC7231].

[RFC2616] and later [RFC7231] Appendix A.5 has text specifically deprecating Content-Transfer-Encoding. However, [RFC7030] incorrectly uses this header.

Any updates to [RFC7030] to bring it inline with HTTP processing risk changing the on-wire protocol in a way that is not backwards compatible. However, reports from implementers suggest that many implementations do not send the Content-Transfer-Encoding, and many of them ignore it. The consequence is that simply deprecating the header would remain compatible with current implementations.

[I-D.ietf-anima-bootstrapping-keyinfra] extends [RFC7030], adding new functionality, and interop testing of the protocol has revealed that unusual processing called out in [RFC7030] causes confusion.

EST is currently specified as part of [IEC62351], and is widely used in Government, Utilities and Financial markets today.

This document therefore revises [RFC7030] to reflect the field reality, deprecating the extraneous field.

This document deals with errata numbers [errata4384], [errata5107], [errata5108], and [errata5904].

This document deals with [errata5107] and [errata5904] in Section 3. [errata5108] is dealt with in Section 5. [errata4384] is closed by correcting the ASN.1 Module in Section 4.

2. 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.

3. Changes to EST endpoint processing

The [RFC7030] sections 4.1.3 (CA Certificates Response, /cacerts), 4.3.1/4.3.2 (Full CMC, /fullcmc), 4.4.2 (Server-Side Key Generation, /serverkeygen), and 4.5.2 (CSR Attributes, /csrattrs) specify the use of base64 encoding with a Content-Transfer-Encoding for requests and response.

This document updates [RFC7030] to require the POST request and payload response of all endpoints use Base64 encoding as specified in Section 4 of [RFC4648]. In both cases, the Distinguished Encoding Rules (DER) [X.690] are used to produce the input for the Base64 encoding routine. This format is to be used regardless of any Content-Transfer-Encoding header, and any value in such a header MUST be ignored.

3.1. Whitespace processing

Note that "base64" as used in the HTTP [RFC2616] does not permit CRLF, while the "base64" used in MIME [RFC2045] does. This specification clarifies that despite [RFC2616], that white space including CR, LF, spaces (ASCII 32) and, tabs (ASCII 9) SHOULD be tolerated by receivers. Senders are not required to insert any kind of white space.

3.2. Changes sections 4 of RFC7030

3.2.1. Section 4.1.3

Replace:

A successful response MUST be a certs-only CMC Simple PKI Response, as defined in [RFC5272], containing the certificates described in the following paragraph. The HTTP content-type of "application/pkcs7-mime" is used. The Simple PKI Response is sent with a Content-Transfer-Encoding of "base64" [RFC2045].

with: (RFCEDITOR: maybe artwork is the wrong choice here)

A successful response MUST be a certs-only CMC Simple PKI Response, as defined in [RFC5272], containing the certificates described in the following paragraph. The HTTP content-type of "application/pkcs7-mime" is used. The CMC Simple PKI Response is encoded in base64 [RFC4648].

3.2.2. Section 4.3.1

Replace:

If the HTTP POST to /fullcmc is not a valid Full PKI Request, the server MUST reject the message. The HTTP content-type used is "application/pkcs7-mime" with an smime-type parameter "CMC-request", as specified in [RFC5273]. The body of the message is the binary value of the encoding of the PKI Request with a Content-Transfer-Encoding of "base64" [RFC2045].

with:

If the HTTP POST to /fullcmc is not a valid Full PKI Request, the server MUST reject the message. The HTTP content-type used is "application/pkcs7-mime" with an smime-type parameter "CMC-request", as specified in [RFC5273]. The body of the message is encoded in base64 [RFC4648].

3.2.3. Section 4.3.2

Replace:

The body of the message is the binary value of the encoding of the PKI Response with a Content-Transfer-Encoding of "base64" [RFC2045].

with:

The body of the message is the base64 [RFC4648] encoding of the PKI Response.

3.2.4. Section 4.4.2

Replace:

An "application/pkcs8"
part consists of the base64-encoded DER-encoded [X.690]
PrivateKeyInfo with a Content-Transfer-Encoding of "base64"
[RFC4648].

with:

An "application/pkcs8" part consists of the base64-encoded DER-encoded [X.690] PrivateKeyInfo.

Replace:

In all three additional encryption cases, the EnvelopedData is returned in the response as an "application/pkcs7-mime" part with an smime-type parameter of "server-generated-key" and a Content-Transfer-Encoding of "base64".

with:

In all three additional encryption cases, the EnvelopedData is returned in the response as an "application/pkcs7-mime" part with an smime-type parameter of "server-generated-key". It is base64 encoded [RFC4648].

3.2.5. Section 4.5.2

This section is updated in its entirety in Section 4.

4. Clarification of ASN.1 for Certificate Attribute set.

Section 4.5.2 of [RFC7030] is to be replaced with the following text:

4.5.2 CSR Attributes Response

If locally configured policy for an authenticated EST client indicates a CSR Attributes Response is to be provided, the server response MUST include an HTTP 200 response code. An HTTP response code of 204 or 404 indicates that a CSR Attributes Response is not available. Regardless of the response code, the EST server and CA MAY reject any subsequent enrollment requests for any reason, e.g., incomplete CSR attributes in the request.

Responses to attribute request messages MUST be encoded as the content-type of "application/csrattrs", and are to be "base64" [RFC4648] encoded. The syntax for application/csrattrs body is as follows:

```
CsrAttrs ::= SEQUENCE SIZE (0..MAX) OF AttrOrOID

AttrOrOID ::= CHOICE {
  oid        OBJECT IDENTIFIER,
  attribute  Attribute {{AttrSet}} }

AttrSet ATTRIBUTE ::= { ... }
```

An EST server includes zero or more OIDs or attributes [RFC2986] that it requests the client to use in the certification request. The client MUST ignore any OID or attribute it does not recognize. When the server encodes CSR Attributes as an empty SEQUENCE, it means that the server has no specific additional information it desires in a client certification request (this is functionally equivalent to an HTTP response code of 204 or 404).

If the CA requires a particular cryptographic algorithm or use of a particular signature scheme (e.g., certification of a public key based on a certain elliptic curve, or signing using a certain hash algorithm) it MUST provide that information in the CSR Attribute Response. If an EST server requires the linking of identity and POP information (see Section 3.5), it MUST include the challengePassword OID in the CSR Attributes Response.

The structure of the CSR Attributes Response SHOULD, to the greatest extent possible, reflect the structure of the CSR it is requesting. Requests to use a particular signature scheme (e.g. using a particular hash function) are represented as an OID to be reflected in the SignatureAlgorithm of the CSR. Requests to use a particular

cryptographic algorithm (e.g., certification of a public key based on a certain elliptic curve) are represented as an attribute, to be reflected as the AlgorithmIdentifier of the SubjectPublicKeyInfo, with a type indicating the algorithm and the values indicating the particular parameters specific to the algorithm. Requests for descriptive information from the client are made by an attribute, to be represented as Attributes of the CSR, with a type indicating the [RFC2985] extensionRequest and the values indicating the particular attributes desired to be included in the resulting certificate's extensions.

The sequence is Distinguished Encoding Rules (DER) encoded [X.690] and then base64 encoded (Section 4 of [RFC4648]). The resulting text forms the application/csrattr body, without headers.

For example, if a CA requests a client to submit a certification request containing the challengePassword (indicating that linking of identity and POP information is requested; see Section 3.5), an extensionRequest with the Media Access Control (MAC) address ([RFC2307]) of the client, and to use the secp384r1 elliptic curve and to sign with the SHA384 hash function. Then, it takes the following:

```
OID:        challengePassword (1.2.840.113549.1.9.7)

Attribute:  type = extensionRequest (1.2.840.113549.1.9.14)
            value = macAddress (1.3.6.1.1.1.1.22)

Attribute:  type = id-ecPublicKey (1.2.840.10045.2.1)
            value = secp384r1 (1.3.132.0.34)

OID:        ecdsaWithSHA384 (1.2.840.10045.4.3.3)
```

and encodes them into an ASN.1 SEQUENCE to produce:

```
30 41 06 09 2a 86 48 86 f7 0d 01 09 07 30 12 06 07 2a 86 48 ce 3d
02 01 31 07 06 05 2b 81 04 00 22 30 16 06 09 2a 86 48 86 f7 0d 01
09 0e 31 09 06 07 2b 06 01 01 01 01 16 06 08 2a 86 48 ce 3d 04 03
03
```

and then base64 encodes the resulting ASN.1 SEQUENCE to produce:

```
MEEGCSqGSIb3DQEJBzASBgcqhkjOPQIBMQcGBSuBBAAiMBYGCSqGSIb3DQEJDjEJ
BgcrBgEBAQEWBggqhkjOPQQDAw==
```

5. Clarification of error messages for certificate enrollment operations

[errata5108] clarifies what format the error messages are to be in. Previously a client might be confused into believing that an error returned with type text/plain was not intended to be an error.

5.1. Updating section 4.2.3: Simple Enroll and Re-enroll Response

Replace:

> If the content-type is not set, the response data MUST be a plaintext human-readable error message containing explanatory information describing why the request was rejected (for example, indicating that CSR attributes are incomplete).

with:

> If the content-type is not set, the response data MUST be a plaintext human-readable error message containing explanatory information describing why the request was rejected (for example, indicating that CSR attributes are incomplete). Servers MAY use the "text/plain" content-type [RFC2046] for human-readable errors.

5.2. Updating section 4.4.2: Server-Side Key Generation Response

Replace:

> If the content-type is not set, the response data MUST be a plaintext human-readable error message.

with:

> If the content-type is not set, the response data MUST be a plaintext human-readable error message. Servers MAY use the "text/plain" content-type [RFC2046] for human-readable errors.

6. Privacy Considerations

This document does not disclose any additional identities to either active or passive observer would see with [RFC7030].

7. Security Considerations

This document clarifies an existing security mechanism. It does not create any new protocol mechanism.

All security considerations from [RFC7030] also apply for the clarifications described in this document.

8. IANA Considerations

The ASN.1 module in Appendix A of this document makes use of object identifiers (OIDs). This document requests that IANA register an OID in the SMI Security for PKIX Arc in the Module identifiers subarc (1.3.6.1.5.5.7.0) for the ASN.1 module. The OID for the Asymmetric Decryption Key Identifier (1.2.840.113549.1.9.16.2.54) was previously defined in [RFC7030].

IANA is requested to update the "Reference" column for the Asymmetric Decryption Key Identifier attribute to also include a reference to this document.

9. Acknowledgements

This work was supported by Huawei Technologies.

The ASN.1 Module was assembled by Russ Housley and formatted by Sean Turner. Russ Housley provided editorial review.

10. References

10.1. Normative References

[errata4384]
"EST errata 4384: ASN.1 encoding error", n.d., <https://www.rfc-editor.org/errata/eid4384>.

[errata5107]
"EST errata 5107: use Content-Transfer-Encoding", n.d., <https://www.rfc-editor.org/errata/eid5107>.

[errata5108]
"EST errata 5108: use of Content-Type for error message", n.d., <https://www.rfc-editor.org/errata/eid5108>.

[errata5904]
"EST errata 5904: use Content-Transfer-Encoding", n.d., <https://www.rfc-editor.org/errata/eid5904>.

[IEC62351]
International Electrotechnical Commission, "Power systems management and associated information exchange - Data and communications security - Part 9: Cyber security key management for power system equipment", ISO/IEC 62351-9:2017, 2017.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

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

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

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, DOI 10.17487/RFC5272, June 2008, <https://www.rfc-editor.org/info/rfc5272>.

[RFC5273] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC): Transport Protocols", RFC 5273, DOI 10.17487/RFC5273, June 2008, <https://www.rfc-editor.org/info/rfc5273>.

[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>.

[RFC6268] Schaad, J. and S. Turner, "Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)", RFC 6268, DOI 10.17487/RFC6268, July 2011, <https://www.rfc-editor.org/info/rfc6268>.

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

[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>.

[X.680] ITU-T, "Information technology - Abstract Syntax Notation One.", ISO/IEC 8824-1:2002, 2002.

[X.681] ITU-T, "Information technology - Abstract Syntax Notation One: Information Object Specification.", ISO/IEC 8824-2:2002, 2002.

[X.682] ITU-T, "Information technology - Abstract Syntax Notation One: Constraint Specification.", ISO/IEC 8824-2:2002, 2002.

[X.683] ITU-T, "Information technology - Abstract Syntax Notation One: Parameterization of ASN.1 Specifications.", ISO/IEC 8824-2:2002, 2002.

[X.690] ITU-T, "Information technology - ASN.1 encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).", ISO/IEC 8825-1:2002, 2002.

10.2. Informative References

[I-D.ietf-anima-bootstrapping-keyinfra] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-43 (work in progress), August 2020.

[RFC2307] Howard, L., "An Approach for Using LDAP as a Network Information Service", RFC 2307, DOI 10.17487/RFC2307, March 1998, <https://www.rfc-editor.org/info/rfc2307>.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[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>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

Appendix A. ASN.1 Module

This annex provides the normative ASN.1 definitions for the structures described in this specification using ASN.1 as defined in [X.680], [X.681], [X.682] and [X.683].

The ASN.1 modules makes imports from the ASN.1 modules in [RFC5912] and [RFC6268].

There is no ASN.1 Module in RFC 7030. This module has been created by combining the lines that are contained in the document body.

```
  PKIXEST-2019
       { iso(1) identified-organization(3) dod(6)
         internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
         id-mod-est-2019(TBD) }

  DEFINITIONS IMPLICIT TAGS ::=
  BEGIN

  -- EXPORTS ALL --

  IMPORTS

  Attribute
  FROM CryptographicMessageSyntax-2010  -- [RFC6268]
        { iso(1) member-body(2) us(840) rsadsi(113549)
          pkcs(1) pkcs-9(9) smime(16) modules(0)
           id-mod-cms-2009(58) }

  ATTRIBUTE
  FROM PKIX-CommonTypes-2009 -- [RFC5912]
      { iso(1) identified-organization(3) dod(6) internet(1) security(5)
        mechanisms(5) pkix(7) id-mod(0) id-mod-pkixCommon-02(57) } ;


  -- CSR Attributes

  CsrAttrs ::= SEQUENCE SIZE (0..MAX) OF AttrOrOID

  AttrOrOID ::= CHOICE {
     oid        OBJECT IDENTIFIER,
     attribute  Attribute {{AttrSet}} }

  AttrSet ATTRIBUTE ::= { ... }


  -- Asymmetric Decrypt Key Identifier Attribute

  aa-asymmDecryptKeyID ATTRIBUTE ::=
      { TYPE AsymmetricDecryptKeyIdentifier
        IDENTIFIED BY id-aa-asymmDecryptKeyID }

  id-aa-asymmDecryptKeyID OBJECT IDENTIFIER ::= { iso(1) member-body(2)
      us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) aa(2) 54 }

  AsymmetricDecryptKeyIdentifier ::= OCTET STRING

  END
```

Authors' Addresses

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

Thomas Werner
Siemens

Email: thomas-werner@siemens.com

Wei Pan
Huawei Technologies

Email: william.panwei@huawei.com