S/MIME Message Specification

PKCS Security Services for MIME

S/MIME Editor

Revised February 23, 1996

1. Overview

This document describes a protocol for adding cryptographic signature and/or encryption services to Internet MIME messages. MIME, described in RFC 1521, provides a general structure for the content type of Internet mail messages and allows extensions for new content type applications. Therefore, this draft defines application/x-pkcs7-mime which specifies that a MIME body part has been cryptographically enhanced according to PKCS #7. This draft also defines application/x-pkcs10 for use in submitting a certification request.

This specification is compatible with PKCS #7 in that it uses the data types defined by PKCS #7. It also inherits all the varieties of architectures for certificate-based key management supported by PKCS #7.

PKCS #7 had the problem that its binary encodings could not be transmitted over 7-bit electronic mail systems. This specification solves that problem by using MIME's content transfer encoding services.

This document discusses the use of multipart/signed and application/x-pkcs7-signature, which can be used to display the "clear text" of a signed message for the benefit of readers using an e-mail system with no security services..

2. References

PKCS #7 RSA Laboratories. *PKCS #7: Cryptographic Message Syntax Standard.* Version 1.5, November 1993.

PKCS #10 RSA Laboratories. *PKCS #10: Certification Request Syntax Standard.* Version 1.0, November 1993.

RFC 1521 N. Borenstein, N. Freed. RFC 1521: MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies. September 1993.

RFC 1847 J. Galvin, S.Murphy, S. Crocker, N. Freed. RFC 1847 Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted. October 1995.

[Conformance] N Freed. IETF draft-ietd-822ext-mime-conf-XX.txt. Multipart Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples. August 1995.

3. Definitions

For the purposes of this draft, the following definitions apply.

ASN.1: Abstract Syntax Notation One, as defined in X.208.

BER: Basic Encoding Rules, as defined in X.209.

`Certificate`: A type that binds an entity's distinguished name to a public key with a digital signature. This type is defined in X.509. This type also contains the distinguished name of the certificate issuer (the signer), an issuer-specific serial number, the issuer's signature algorithm identifier, and a validity period.

`CertificateRevocationList (CRL)`: A type that contains information about certificates whose validity an issuer has prematurely revoked. The information consists of an issuer name, the time of issue, the next scheduled time of issue, and a list of certificate serial numbers and their associated revocation times. The CRL is signed by the issuer. The type intended by this standard is the one defined RFC 1422.

DER: Distinguished Encoding Rules for ASN.1, as defined in X.509, Section 8.7.

MIME: Multipurpose Internet Mail Extensions, as defined in RFC 1521.

Data Types: These definitions are those used with the MIME and SMTP standards.

- **7-bit**: Text data with lines less than 998 characters long, no characters with the 8th bit set and no NULL characters. <CR> and <LF> occur only as part of a <CR><LF> end of line delimiter.

- **8-bit**: Text data with lines less than 998 characters, and no NULL characters. <CR> and <LF> occur only as part of a <CR><LF> end of line delimiter.

- **binary**: Arbitrary data.

Transfer Encoding: A reversible transformation made on data so 8-bit or binary data may be sent via a channel that only transmits 7-bit data.

4. Content-Type application/x-pkcs7-mime

This chapter defines the format of data used in application/x-pkcs7-mime. First we introduce the data format and discuss various implementation issues. Then we describe the steps an agent must follow to send or receive an application/x-pkcs7-mime body part. Finally, we discuss some security considerations.

4.1 Format of the application/x-pkcs7-mime body

PKCS #7 defines a general ASN.1 type, ContentInfo, for use in exchanging cryptographic information. For convenience, its definition is repeated here:

```
ContentInfo ::= SEQUENCE {
  contentType ContentType,
  content
    [0] EXPLICIT ANY DEFINED BY contentType OPTIONAL }

ContentType ::= OBJECT IDENTIFIER
```

PKCS #7 also defines several content types which can be used within a ContentInfo. For our purposes here, the most important are SignedData (for exchanging digitally signed data), EnvelopedData (for exchanging digitally enveloped data) and SignedAndEnvelopedData (for exchanging data both digitally signed and enveloped).

Therefore, when the MIME content type application/x-pkcs7-mime is used, the body shall be a ContentInfo as defined by PKCS #7, encoded using the Basic Encoding Rules (BER). The PKCS #7 content type will normally be SignedData, EnvelopedData or SignedAndEnvelopedData, but use of other content types defined by PKCS #7 are not disallowed.

Notes.

- Since BER is specified, instead of the more restrictive DER, an application may use techniques such as indefinite-length encoding. This is especially useful for transferring large data or streamed data where the total length is not known in advance.

- Data produced by BER is 8-bit, but most mailers require 7-bit data. Therefore, a suitable 7-bit Content-Transfer-Encoding is needed, such as base64. This is discussed in a later section.

- PKCS #7 has a provision for "detached data", where, for example, the SignedData in the ContentInfo contains only the signature information, but not the actual data which is signed (which is transferred separately). Application/x-pkcs7-mime explicitly disallows the use of detached data and requires the data to be present within the ContentInfo. The “detached data” provision is, however, used by the multipart/signed construct described in Section 6.

- PKCS #7 provides for a degenerate case of SignedData which can be used for disseminating certificates and CRLs. This is explicitly allowed in application/x-pkcs7-mime. In this case, the content is omitted from the ContentInfo and there are no entries in the SignerInfos, leaving only entries for certificates and crls. Typically, this is used to convey certificates in response to a certification request, or CRLs in response to a CRL retrieval request. The MIME agent should process and/or store the certificates and CRLs and may choose to display a confirmation to the user.

4.2 Format of the signed or enveloped data

PKCS #7 places no requirements on the format of the data which is signed or enveloped. However, for use in application/x-pkcs7-mime, the signed or enveloped data must itself be a MIME entity. Therefore, when a MIME agent receives an application/x-pkcs7-mime, the result of removing the signature or envelope can be passed directly to the normal MIME-processing software.

Because the MIME entity being signed or enveloped is likely to be transferred to and processed on a different platform than it was created it is important that it be in canonical or “on-the-wire” format. The most common difference between platforms is the end of line delimiter. The use of the canonical format is most important for signing messages. In addition to being in canonical MIME format there are some considerations for the selection of the transfer encoding for the MIME entity that affect the ability to verify the signature of a received MIME entity, especially if it is to be subsequently forwarded. Since the data which is signed or enveloped will be recovered by the recipient in the exact form it was produced by the sender, we must take care that this data be in a canonical format which can be processed by any platform.

A rough outline of what is needed to convert a MIME entity into canonical format is given here, but the implementor is referred to the MIME RFCs for complete details. It is also advisable to check for IETF drafts of the MIME specification that are later than the most recent RFCs until MIME is published as a complete IETF standard. This following set of steps is prescriptive to illustrate exactly what the end result must be. An implementation need not perform these steps exactly as long as the end result is the same as if the steps were performed.

Given that the data objects to be included in the message are available in the local format, the first step is to convert each object to its canonical format. The canonical format of each object is determined by its MIME type and subtype as it is tagged in the MIME entity. For example, data of MIME type text/plain must have each end of line delimiter converted to <CR><LF>, and an isolated <CR> or <LF> is not permitted. Another canonicalization that may be required is the conversion from the local character set to a standard character set. Many data types such as image/gif or application/postscript have only one format and this serves as both the canonical format and the local format no matter the platform.

The second step is the application of transfer encoding. Transfer encoding is used to transform 8-bit text (e.g., in the ISO-8859-1 character set) or binary data (arbitrary line length, may include any octet including NULL) in such a way that it can be transmitted over 7-bit or non-binary channels. Note that when transfer encoding is applied in this step the output lines of the transfer encoder will be lines of 7-bit text with canonical line ending. Any transfer encoding to the MIME entity inside the PKCS envelope will be referred to as the “inside transfer encoding." Strictly speaking no transfer encoding is required for a MIME object inside the PKCS envelope since it will be embedded in a PKCS object which is effectively a binary channel, but there are some further considerations. If the PKCS envelope is ever removed and the resulting MIME object is ever to be transmitted by itself without further processing via standard SMTP transport it is advisable to use some transfer encoding. The choice is at the discretion of the implementor. If no Content-Transfer-Encoding header is included, it is assumed that the transfer encoding is 7-bit and that the data will be conformant. If the data is binary, then a Content-Transfer-Encoding: binary header must be included. The same is true for 8-bit data.

Third, the encoded sub-entities are inserted in the full MIME entity. This step consists essentially of concatenating the sub-entities interspersed with the appropriate MIME Content-type, Content-Transfer-Encoding and related headers and part boundaries. Note that MIME headers and boundaries are all text with canonical line endings.

Notes.

- The requirement that the signed or enveloped data must itself be a MIME body part is why the protocol defined here is called application/x-pkcs7-mime and not just application/x-pkcs7. A protocol for transferring a PKCS #7 object with arbitrary signed or enveloped content could be defined, but is outside the scope of this document.

4.3 Outside Content transfer encoding

The body of application/x-pkcs7-mime is a BER-encoded PKCS #7 ContentInfo. However, as mentioned above, the result of BER-encoding is binary data, not 7-bit text as required by most SMTP mail transport agents. In general, base64 Content-Transfer-Encoding can be used, though all that is required is that the transfer encoding be such that application/x-pkcs7-mime entity can be transferred intact. (It is even possible that the transfer encoding will be changed by the message transfer agent, as this is explicitly permitted.)

4.4 Sending an application/x-pkcs7-mime body part

Now that we have introduced the various formats and issues, we can describe the steps for sending a message using application/x-pkcs7-mime. In this example, we create a digital envelope, but the same process applies to other PKCS #7 content types. Note that the first four steps listed here are the steps for preparing a MIME entity for transmission via SMTP with the exception that binary transfer encoding is allowed. The implementor is referred to the MIME conformance document for the precise details. Also, these steps need not be performed exactly as described here as long as the result obtained is identical.

- Each message part is prepared as usual by the message composition software and becomes available in its local format.

- Each message part is converted to its canonical format as defined for its MIME content type (e.g., end of line delimiters are converted to <CR><LF> for type text/plain).

- If desired by the implementor, transfer encoding is applied to each message part. It is optional as the default inside transfer encoding for message parts is binary. If the data for a MIME part is not 7-bit data, then a Content-Transfer-Encoding header indicating whether it is either 8-bit or binary **must** be inserted as part of the following step.

- MIME Content-xxxx headers are added to each part, and if there are multiple parts each entity is inserted into a multipart message. That is, the message parts are assembled into a complete MIME entity.

- The MIME entity is encrypted according to PKCS #7 EnvelopedData, where the encrypted message is placed in the EnvelopedData's EncryptedContentInfo. (Note that the EnvelopedData also contains the RecipientInfo data which the recipient will use to open the envelope and decrypt the message.)

- The EnvelopedData is placed within a PKCS #7 ContentInfo.

- The ContentInfo is encoded according to BER.

- As a typical step, the BER-encoded ContentInfo is also base64 encoded so that it is 7-bit data suitable for mail transfer agents.

- This then becomes the body of an application/x-pkcs7-mime body part. The result might look like this:

```
Content-Type: application/x-pkcs7-mime
Content-Transfer-Encoding: base64
```


```
ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756
```


4.5 Receiving an application/x-pkcs7-mime body part

These are the steps for receiving the application/x-pkcs7-mime message from the previous example:

- The content transfer encoding is removed, which in this example is base64.

- The PKCS #7 ContentInfo is examined to determine the PKCS #7 content type, which in this case is EnvelopedData.

- The EnvelopedData is processed in order to decrypt the content, which must be a MIME body part as required by application/x-pkcs7-mime.

- The result is passed to a standard MIME processor which removes transfer encoding, splits multipart entities, and possibly converts the parts to local format. In this example, the message "This is an encrypted message" is displayed to the user.

4.6 Security considerations

All of the security issues faced by any cryptographic application must be faced by a MIME agent which supports application/x-pkcs7-mime. Among these are protecting the user's private key, preventing various attacks, and helping the user avoid mistakes such as inadvertently encrypting a message for the wrong recipient. These issues are beyond the scope of this document, however we can make the following notes:

- An implementor may choose not to present data which has been digitally signed if the signature verification fails. The reasoning is that if the signature fails, the data has been tampered with and should be considered untrustworthy. This is consistent with the fact that the signed data within an application/x-pkcs7-mime entity cannot be seen by viewing the "raw" MIME message, since it is base64 encoded (unless is it sent clear signed as described in section 6). The message must be processed and the signature verified before the data is presented. If the implementor does choose to present the data even if the signature verification fails, it is advisable to strongly warn the user.

- A MIME agent which supports application/x-pkcs7-mime inherits the certificate-based key management supported by PKCS #7. In particular, a typical agent must be able to process certificates and CRLs which may be included inside a SignedData or SignedAndEnvelopedData type. The agent may also include auxiliary services to allow the user to generate keys, submit and process certification requests and to request updated CRLs.

5. Content-Type application/x-pkcs10

A typical application which allows a user to generate cryptographic information must submit that information to a certification authority, who transforms it into a certificate. PKCS #10 describes a syntax for certification requests. We therefore define application/x-pkcs10 which can be used to transfer a PKCS #10 certification request.

The details of certification requests and the process of obtaining a certificate are beyond the scope of this document. Here, we only define the format of data used in application/x-pkcs10. First we introduce the data format and discuss various implementation issues. Then we describe the steps an agent must follow to send or receive an application/x-pkcs10 body part.

5.1 Format of the application/x-pkcs10 body

PKCS #10 defines the ASN.1 type CertificationRequest for use in submitting a certification request. Therefore, when the MIME content type application/x-pkcs10 is used, the body shall be a CertificationRequest, encoded using the Basic Encoding Rules (BER).

Notes.

- Although BER is specified, instead of the more restrictive DER, a typical application will use DER since the CertificationRequest's CertificationRequestInfo must be DER-encoded anyway in order to be signed. A robust application should output DER, but allow BER or DER on input.

- Data produced by BER or DER is 8-bit, but most mailers require 7-bit data. Therefore, a suitable 7-bit Content-Transfer-Encoding is needed. This document recommends using the base64 Content-Transfer-Encoding with application/x-pkcs10, although any 7-bit transfer encoding will work.

5.2 Sending and receiving an application/x-pkcs10 body part

Now that we have introduced the various formats and issues, we can describe the steps for sending a message using application/x-pkcs10.

- The application generates the cryptographic information for the user. The details of this are beyond the scope of this document.
- The cryptographic information is placed within a PKCS #10 CertificationRequest.
- The CertificationRequest is encoded according to BER or DER (typically, DER).
- As a typical step, the DER-encoded CertificationRequest is also base64 encoded so that it is 7-bit data suitable for mail transfer agents.
- This then becomes the body of an application/x-pkcs10 body part. The result might look like this:

```
Content-Type: application/x-pkcs10
Content-Transfer-Encoding: base64

rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGfHfQbnj756YT64V
```

A typical application only needs to send a certification request. It is a certification authority which must receive and process the request. The steps for recovering the CertificationRequest from the message are straightforward and are not presented here. The procedures for processing the certification request are beyond the scope of this document.

6. Use of the Multipart/Signed Content-Type

RFC 1847 defines the multipart/signed content type explicitly for clear-signed messages. Clear signing means the content is in the clear so that recipients with older e-mail systems that cannot verify the signatures can still read the message even if they cannot verify the signature. The multipart/signed MIME type is a multipart type with exactly two parts. The first part carries the MIME entity being signed. The second part carries the signature.

The multipart/signed Content type has the following parameters:

(1) The protocol parameter (required): “application/x-pkcs7-signature”

(2) The micalg parameter: rsa-md5

Note that quoting is required around the protocol parameter as a “/” character in a parameter value must be quoted.

The micalg parameter allows for one-pass processing when the signature is being verified. The value of the micalg parameter is dependent on the message digest algorithm used in the calculation of the Message Integrity Check, rsa-md5 is included as an example. This specification does not require or recommend any particular choice of mic algorithm.

6.1 Preparation of the data for signing

The data is prepared as described in section 4 with the following extra considerations for transfer encoding. All parts of the MIME entity must have transfer encoding of 7-bit, quoted-printable, or base64. No binary or 8-bit data is permitted. This is required because the current Internet mail transport infrastructure cannot guarantee transport of unencoded binary or 8-bit data and because the digest for the signature is computed on the fully encoded message. If a message with 8-bit data were to encounter a transfer agent that can not transmit 8-bit or binary data it has three options, none of which are acceptable for a clear-signed message. First, it could change the transfer encoding, but that would invalidate the signature. It could transmit anyway which would most likely result in the 8th bit being stripped, also resulting in the signature being invalidated. Last it could return the message to the sender. Thus transfer encoding to make the data 7-bit text is required before computing the signature. Note RFC 1847 prohibits an MTA from changing the transfer encoding of the first part of a multipart/signed message. If an MTA encountered a multipart/signed message with 8-bit or binary data in the first part, it would have to return the message to the sender as undeliverable.

Note that is it is advisable to use quoted printable transfer encoding in some situations that may not otherwise seem necessary to better preserve the data so the signature can be verified. For example, trailing spaces should always be quoted-printable encoded as some mail systems strip them if they are not transferred with encoding. Also the word “From” followed by a space at the beginning of a line should be encoded as some mail systems convert this to ">From". Because of the restriction on 8-bit transport it is also necessary to use quoted printable encoding for text using an 8-bit character set such as ISO-8859-1.

6.2 The application/x-pkcs7-signature type

The second part of the multipart/signed message must have type application/x-pkcs7-signature. The data in the second part is the PKCS #7 detached signature. A detached signature is a PKCS #7 data object of type `SignedData` with the `signerInfos` field containing signatures on some external data and the `ContentInfo` field empty. In this case the external data is the fully encoded MIME entity described above and placed in the first part of the multipart/signed message.

6.3 Procedure for sending a clear signed message

- The data is prepared as described above so the transfer encoding of all the parts is either 7-bit, quoted-printable or base64.
- The PKCS #7 detached signature for the data is computed.
- The content data is inserted into the first part of the multipart/signed MIME entity.
- The detached signature is inserted into the second part of the multipart/signed message using appropriate transfer encoding (most likely base64). The MIME type of the detached signature part must be application/x-pkcs7-signature.

6.4 Procedure for receiving a clear signed message

- The message digest of the first part of the multipart/signed message is computed before any further MIME decoding or processing other than separating the two parts of the multipart/signed entity .
- The application/x-pkcs7-signature information in the second part is processed along with the digest obtained in the first step to verify the signature.
- The MIME processing and decoding of the first part proceeds and the message is presented to the user along with the result of the signature verification.

6.5 Example multipart/signed message

```
Content-Type: multipart/signed;
  protocol="application/x-pkcs7-signature";
  micalg=rsa-md5; boundary=boundary42

--boundary42
Content-Type: text/plain

This is a clear-signed message

--boundary42
Content-Type: application/x-pkcs7-signature
Content-Transfer-Encoding: base64

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--boundary42--
```

7. Author's address

S/MIME Editor
RSA Data Security, Inc.
100 Marine Parkway, #500
Redwood City, CA 94065 USA
(415) 595-8782
(415) 595-1873 (fax)
smime-editor@rsa.com

