PKCS Security Services for MIME ("S/MIME")

PKCS Steering Group
PRELIMINARY DRAFT
Revised April 19, 1995

1. Overview

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

PKCS #7 had the problem that its 8-bit encodings could not be transmitted over 7-bit electronic mail systems. This draft solves that problem by using MIME's content transfer encoding services.

This document discusses the use of multipart/alternative, which can be used to display the "clear text" of a signed message for the benefit of non-MIME readers.

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 Syntx Standard.* Version 1.0, November 1993.

DRAFT - DRAFT - Copyright © 1995 RSA Data Security, Inc. - DRAFT

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.

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). However, application/x-pkcs7-mime explicitly disallows the use of detached data and requires the data to be present within the ContentInfo.

- 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 body part. 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.

The MIME body part contains end-of-line delimiters. However, various platforms use different native end-of-line delimiters. 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.

Therefore, before a MIME body part is signed or enveloped, its end-line-line delimiters must be canonicalized to <CR><LF>. If the body is 7-bit, then the entire body part, including the header and body, is canonicalized to <CR><LF>. (This includes the case where a content transfer encoding has been used, since the data being signed or enveloped is 7-bit, even though the original data may have been 8-bit before the transfer encoding was applied.)

However, if the body is 8-bit, and no content transfer encoding is used, then only the end-of-line delimiters in the header and the following blank line are canonicalized to <CR><LF>. The data in the body is not altered.

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 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 8-bit data, not 7-bit as required by most mailers. Fortunately, MIME has defined the Content-Transfer-Encoding service to solve this type of problem. Therefore, this document recommends using the base64 Content-Transfer-Encoding with application/x-pkcs7-mime, although any 7-bit transfer encoding will work.

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.

- The agent prepares a standard MIME body part, such as the following. (Of course, it could be any MIME type, including multipart.)

```
Content-Type: text/plain; charset="us-ascii"

This is an encrypted message
```

- The message is canonicalized to have <CR><LF> end-of-line delimiters. Since the body of this message is 7-bit, not 8-bit, both the headers and the body are recoded with <CR><LF> end-of-line delimiters.

- The message 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 canonical <CR><LF> end-of-line delimiters are converted to the native format used by the MIME processor. Since the body of this message is 7-bit, not 8-bit, the the <CR><LF> delimiters in the body as well as the header are converted to the native format.

- The result is passed to the MIME processor. 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 cannot be seen by viewing the "raw" MIME message, since it is base64 encoded. 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 certifiation 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 multipart/alternative for showing clear text

When a simple text message is digitally signed according to application/x-pkcs7-mime, it is placed within a BER-encoded PKCS #7 ContentInfo which is typically encoded with the base64 content transfer encoding. The result is that the original simple text message cannot be seen if the application/x-pkcs7-mime message is viewed in its "raw" form.

A user with a "MIME-aware" agent will seldom see the message in its raw form. However, many users still do not have MIME-aware agents, and so there may be a desire to add a digital signature to a text message while also showing the text "in the clear". Therefore, we discuss the use of multipart/alternative for displaying the clear text of a signed message along with the application/x-pkcs7-mime body part.

6.1 Format of multipart/alternative with application/x-pkcs7-mime

RFC 1521 defines multipart/alternative as a means of transferring the same message in multiple formats of different complexity so that the recipient's mail reader can display the best form it is capable of processing.

Suppose the text message to be digitally signed is:

```
Content-Type: text/plain; charset="us-ascii"

This is a signed message
```

And suppose the result of signing this by transforming it into an application/x-pkcs7-mime message is:

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

jhJhjH77n8HHGghyHhHUuTrfvbnj756tbB9HG4VQpIGfHfYT6fyF467Gh
I8HHGTrf vhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
IGfHfY7GhT64VQbnj756vhJhjH77
```

After being formed into a multipart/alternative, the message would look something like this:

```
Content-Type: multipart/alternative; boundary=boundary42

--boundary42
Content-Type: text/plain; charset="us-ascii"

This is a signed message

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

jhJhjH77n8HHGghyHhHUuTrfvbnj756tbB9HG4VQpIGfHfYT6fyF467Gh
I8HHGTrf vhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
IGfHfY7GhT64VQbnj756vhJhjH77

--boundary42
```

Notes:

- A user viewing this message in a non-MIME-conformant reader can see the original message within the "raw" MIME message.

- A MIME reader should display the last of the alternative parts of the multipart/alternative which it is capable of processing. Therefore, a MIME reader which recognizes application/x-pkcs7-mime will ignore the "clear text" in the first body part and process the digitally signed message in the second body part.

6.2 Discussion

An issue with supplying the original message text in the clear is this: we produce a clear-text signature to allow a user to view a message without processing it. However, this may be contrary to the purpose of the digital signature because the message is not to be trusted unless the signature has been verified. One cannot expect the casual viewer of this "raw" multipart/alternative message to infer that a digital signature is involved or to understand what that means.

A possible disadvantage of the multipart/alternative approach is that a second copy of the full text of the original message must be included, which might be a problem if the text is long. But note that this technique is only meant for allowing users to view simple text messages and would not be used for large data such as binary files or audio data.

7. Author's address

PKCS Steering Group
RSA Data Security, Inc.
100 Marine Parkway, #500
Redwood City, CA 94065 USA

(415) 595-8782
(415) 595-1873 (fax)
pkcs-editor@rsa.com