        Securing X.400 Content with Secure/Multipurpose Internet Mail
                             Extensions (S/MIME)

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Abstract

   This document describes a protocol for adding cryptographic signature
   and encryption services to X.400 content with Secure/Multipurpose
   Internet Mail Extensions (S/MIME).

1. Introduction

   The techniques described in the Cryptographic Message Syntax [CMS]
   specification are general enough to support many different content
   types. The [CMS] specification thus provides many options for
   providing different security mechanisms. In order to ensure
   interoperability of systems within the X.400 community, it is
   necessary to specify the use of CMS features to protect X.400 content
   (called "CMS-X.400" in this document).

1.1. Specification Overview

   This document is intended to be similar to the S/MIME Version 3.1
   Message Specification [MSG] except that it is tailored to the
   requirements of X.400 content rather than Multipurpose Internet Mail
   Extensions (MIME).

This document defines how to create an X.400 content type that has
been cryptographically enhanced according to [CMS]. In order to
create S/MIME messages carrying X.400 content, an S/MIME agent has to
follow specifications in this document, as well as the specifications
listed in [CMS]. This memo also defines new parameter values for the
application/pkcs7-mime MIME type that can be used to transport those
body parts.

Throughout this document, there are requirements and recommendations
made for how receiving agents handle incoming messages. There are
separate requirements and recommendations for how sending agents
create outgoing messages. In general, the best strategy is to "be
liberal in what you receive and conservative in what you send". Most
of the requirements are placed on the handling of incoming messages
while the recommendations are mostly on the creation of outgoing
messages.

This document does not address transport of CMS-X.400 content. It is
assumed that CMS-X.400 content would be transported by Internet mail
systems, X.400, or other suitable transport.

This document describes applying security services to the content of
entire X.400 messages, which may or may not be IPMS messages. These
objects can be carried by several means, including SMTP-based mail
and X.400 mail. Note that cooperating S/MIME agents must support
common forms of message content in order to achieve interoperability.

If the CMS objects are sent as parts of an RFC 822 message, a
standard MIXER gateway [MIXER] will most likely choose to encapsulate
the message. This is not likely to be a format that is usable by an
X.400 recipient. MIXER is specifically focused on translation
between X.420 Interpersonal Messages and non-secure RFC822/MIME
messages. The discussion of security-related body parts in sections
7.3 and 7.4 of [BODYMAP] is relevant to CMS messages.

Definition of gateway services to support relay of CMS object between
X.400 and SMTP environments is beyond the scope of this document.

1.2. Terminology

The key words "MUST", "SHALL", "REQUIRED", "SHOULD", "RECOMMENDED",
and "MAY" in this document are to be interpreted as described in BCP
14, RFC 2119 [MUSTSHOULD].

1.3. Definitions

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

ASN.1:   Abstract Syntax Notation One, as defined in
                    ISO/IEC 8824.

BER: Basic Encoding Rules for ASN.1, as defined in
                    ISO/IEC 8825-1.

Certificate: A type that binds an entity's distinguished name
                    to a public key with a digital signature.

DER: Distinguished Encoding Rules for ASN.1, as defined
                    in ISO/IEC 8825-1.

7-bit data: Text data with lines less than 998 characters
                    long, where none of the characters have the 8th
                    bit set, and there are no NULL characters. <CR>
                    and <LF> occur only as part of a <CR><LF> end of
                    line delimiter.

8-bit data: Text data with lines less than 998 characters, and
                    where none of the characters are NULL characters.
                    <CR> and <LF> occur only as part of a <CR><LF> end
                    of line delimiter.

Binary data: 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.

Receiving agent: Software that interprets and processes S/MIME CMS
                    objects.

Sending agent: Software that creates S/MIME CMS objects.

S/MIME agent: User software that is a receiving agent, a sending
                    agent, or both.

1.4. Compatibility with Prior Practice of S/MIME

There are believed to be no existing X.400 implementations that
support S/MIME version 2. Further, signed interoperability between
X.400 and MIME systems that support S/MIME version 2 is not believed

to be easily achievable. Therefore backward compatibility with
S/MIME version 2 is not considered to be a requirement for this
document.

It is a goal of this document to, if possible, maintain backward
compatibility with existing X.400 implementations that employ S/MIME
v3.1 wrappers.

2. CMS Options

CMS allows for a wide variety of options in content and algorithm
support. This section puts forth a number of support requirements
and recommendations in order to achieve a base level of
interoperability among all CMS-X.400 implementations. [CMS] provides
additional details regarding the use of the cryptographic algorithms.

2.1. DigestAlgorithmIdentifier

Sending and receiving agents MUST support SHA-1 [CMSALG].

2.2. SignatureAlgorithmIdentifier

Receiving agents MUST support id-dsa-with-sha1 defined in [CMSALG].
The algorithm parameters MUST be absent (not encoded as NULL).
Receiving agents MUST support rsaEncryption, defined in [CMSALG].

Sending agents MUST support either id-dsa-with-sha1 or rsaEncryption.

2.3. KeyEncryptionAlgorithmIdentifier

Sending and receiving agents MUST support rsaEncryption, defined in
[CMSALG].

Sending and receiving agents SHOULD support Diffie-Hellman defined in
[CMSALG].

2.4. General Syntax

The general syntax of CMS objects consist of an instance of the
ContentInfo structure containing one of several defined CMS content
types. CMS defines multiple content types. Of these, only the
SignedData and EnvelopedData content types are used for CMS-X.400.

2.4.1. SignedData Content Type

Sending agents MUST use the signedData content type to apply a
digital signature to a message or, in a degenerate case where there
is no signature information, to convey certificates.

2.4.2. **EnvelopedData Content Type**

   Senders MUST use the envelopedData content type to apply privacy
   protection to a message. A sender needs to have access to a public
   key for each intended message recipient to use this service. This
   content type does not provide authentication.

2.5. **Attribute SignerInfo Type**

   The SignerInfo type allows the inclusion of unsigned and signed
   attributes to be included along with a signature.

   Receiving agents MUST be able to handle zero or one instance of each
   of the signed attributes listed here. Sending agents SHOULD generate
   one instance of each of the following signed attributes in each CMS-
   X400 message:

   - signingTime
   - sMIMECapabilities
   - sMIMEEncryptionKeyPreference

   Requirements for processing of these attributes MUST be in accordance
   with the S/MIME Message Specification [MSG]. Handling of the
   signingTime attribute MUST comply with clause 2.5.1 of [MSG].
   Handling of the sMIMECapabilities attribute MUST comply with clause
   2.5.2 of [MSG]. Handling of the sMIMEEncryptionKeyPreference
   attribute MUST comply with clause 2.5.3 of [MSG].

   Further, receiving agents SHOULD be able to handle zero or one
   instance in the signed attributes of the signingCertificate attribute
   [ESS].

   Sending agents SHOULD generate one instance of the signingCertificate
   signed attribute in each CMS-X400 message.

   Additional attributes and values for these attributes may be defined
   in the future. Receiving agents SHOULD handle attributes or values
   that they do not recognize in a graceful manner.

   Sending agents that include signed attributes that are not listed
   here SHOULD display those attributes to the user, so that the user is
   aware of all of the data being signed.

2.6. **ContentEncryptionAlgorithmIdentifier**

   Sending and receiving agents MUST support encryption and decryption
   with DES EDE3 CBC, hereinafter called "tripleDES" [CMSALG]. Sending
   and receiving agents SHOULD support encryption and decryption with
   AES [CMSAES] at a key size of 128, 192 and 256 bits.

3. **Creating S/MIME Messages**

   This section describes the S/MIME message formats and how they can be
   used to secure X.400 contents. The S/MIME messages are a combination
   of X.400 contents and CMS objects (i.e., a ContentInfo structure
   containing one of the CMS-defined content types). The X.400 content
   and other data, such as certificates and algorithm identifiers, are
   given to CMS processing facilities which produces a CMS object. This
   document also describes how nested, secured S/MIME messages should be
   formatted when encapsulating an X.400 content, and provides an
   example of how a triple-wrapped S/MIME message over X.400 content
   should be created if backwards compatibility with S/MIME version 2 is
   of no concern.

   S/MIME provides one format for enveloped-only data, several formats
   for signed-only data, and several formats for signed and enveloped
   data. The different formats are required to accommodate several
   environments, in particular for signed messages. Only one of these
   signed formats is applicable to X.400.

   Note that canonicalization is not required for X.400 content because
   it is a binary rather than text encoding, and only the "embedded"
   content version is used. These dramatically simplify the description
   of S/MIME productions.

   The reader of this section is expected to understand X.400 as
   described in [X.400] and S/MIME as described in [CMS] and [ESS].

3.1. **The X.400 Message Structure**

   This section reviews the X.400 message format. An X.400 message has
   two parts, the envelope and the content, as described in X.402
   [X.400]:

   Envelope -- An information object whose composition varies from one
   transmittal step to another and that variously identifies the
   message's originator and potential recipients, documents its previous
   conveyance and directs its subsequent conveyance by the Message
   Transfer System (MTS), and characterizes its content.

Content -- The content is the piece of information that the
originating User Agent wants to be delivered to one or more
recipients. The MTS neither examines nor modifies the content,
except for conversion, during its conveyance of the message. MTS
conversion is not applicable to the scenario of this document because
such conversion is incompatible with CMS protection mechanisms.

One piece of information borne by the envelope identifies the type of
the content. The content type is an identifier (an ASN.1 OID or
Integer) that denotes the syntax and semantics of the content
overall. This identifier enables the MTS to determine the message's
deliverability to particular users, and enables User Agents and
Message Stores to interpret and process the content.

Another piece of information borne by the envelope identifies the
types of encoded information represented in the content. An encoded
information type (EIT) is an identifier (an ASN.1 Object Identifier
or Integer) that denotes the medium and format (e.g., IA5 text or
Group 3 facsimile) of individual portions of the content. It further
enables the MTS to determine the message's deliverability to
particular users, and to identify opportunities for it to make the
message deliverable by converting a portion of the content from one
EIT to another.

This document describes how S/MIME CMS is used to secure the content
part of X.400 messages.

3.2. Creating a Signed-only Message with X.400 Content

The SignedData format as described in the Cryptographic Message
Syntax [CMS] MUST be used for signing of X.400 contents.

The X.400 content to be protected MUST be placed in the SignedData
encapContentInfo eContent field. Note that this X.400 content SHOULD
maintain the encoding defined by the content type, but SHOULD NOT be
MIME wrapped. The object identifier for the content type of the
protected X.400 content MUST be placed in the SignedData
encapContentInfo eContentType field.

The signedData object is encapsulated by a ContentInfo SEQUENCE with
a contentType of id-signedData.

Note that if SMTP [SMTP] is used to transport the resulting signed-
only message then the optional MIME encoding SHOULD be used. If
binary transports such as X.400 are used then the optional MIME
encoding SHOULD NOT be used.

There are many reasons for this requirement. An outer MIME wrapper
should not be used in X.400. Further, there are places where X.400
systems will interact with SMTP/MIME systems where the outer MIME
wrapper might be necessary. Because this wrapping is outside the
security wrappers, any gateway system that might bridge the gap
between the two systems will be smart enough to apply or remove the
outer MIME wrapper as appropriate.

3.2.1. MIME Wrapping to Dynamically Support 7-bit Transport

The signedData object MAY optionally be wrapped in MIME. This allows
the system to support 7-bit transport when required. This outer MIME
wrapper MAY be dynamically added or removed throughout the delivery
path since it is outside the signature and encryption wrappers. In
this case the application/pkcs7-mime type as defined in S/MIME
Version 3.1 Message Specification [MSG] SHOULD be used with the
following parameters:

Content-Type: application/pkcs7-mime; smime-type=signed-x400
Content-Transfer-Encoding: base64

If the application/pkcs7-mime MIME type is used to support 7-bit
transport, the steps to create this format are:

Step 1. The X.400 content to be signed is ASN.1 encoded.

Step 2. The ASN.1 encoded X.400 content and other required data is
processed into a CMS object of type SignedData. The SignedData
structure is encapsulated by a ContentInfo SEQUENCE with a
contentType of id-signedData.

Step 3. The CMS object is inserted into an application/pkcs7-mime
MIME entity.

The smime-type parameter for messages using application/pkcs7-mime
with SignedData is "signed-x400" as defined in [TRANSPORT].

3.3. Creating an Enveloped-only Message with X.400 Content

This section describes the format for enveloping an X.400 content
without signing it. It is important to note that sending enveloped
but not signed messages does not provide for data integrity. It is
possible to replace ciphertext in such a way that the processed
message will still be valid, but the meaning is altered.

The EnvelopedData format as described in [CMS] is used for
confidentiality of the X.400 contents.

The X.400 content to be protected MUST be placed in the EnvelopedData
encryptedContentInfo encryptedContent field. Note that this X.400
content SHOULD maintain the encoding defined by the content type, but
SHOULD NOT be MIME wrapped. The object identifier for content type
of the protected X.400 content MUST be placed in the EnvelopedData
encryptedContentInfo contentType field.

The envelopedData object is encapsulated by a ContentInfo SEQUENCE
with a contentType of id-envelopedData.

Note that if SMTP is used to transport the resulting enveloped-only
message then the optional MIME encoding SHOULD be used. If other
transport (e.g., X.400) that is optimized for binary content is used
then the optional MIME encoding SHOULD NOT be used.

3.3.1. MIME Wrapping to Dynamically Support 7-bits Transport

The envelopedData object MAY optionally be wrapped in MIME. This
allows the system to support 7-bit transport when required. This
outer MIME wrapper MAY be dynamically added or removed throughout the
delivery path since it is outside the signature and encryption
wrappers. In this case, the application/pkcs7-mime type as defined
in S/MIME Version 3.1 Message Specification [MSG] SHOULD be used with
the following parameters:

Content-Type: application/pkcs7-mime; smime-type=enveloped-x400
Content-Transfer-Encoding: base64

If the application/pkcs7-mime MIME type is used to support 7-bit
transport, the steps to create this format are:

Step 1. The X.400 content to be enveloped is ASN.1 encoded.

Step 2. The ASN.1 encoded X.400 content and other required data is
processed into a CMS object of type EnvelopedData. In addition to
encrypting a copy of the content-encryption key for each recipient, a
copy of the content encryption key SHOULD be encrypted for the
originator and included in the EnvelopedData (see [CMS] Section 6).
The EnvelopedData structure is encapsulated by a ContentInfo SEQUENCE
with a contentType of id-envelopedData.

Step 3. The CMS object is inserted into an application/pkcs7-mime
MIME entity to allow for 7-bit transport.

If the application/pkcs7-mime MIME entity is used, the smime-type
parameter for enveloped-only messages is "enveloped-x400" as defined
in [TRANSPORT].

3.4. Nested CMS Structures

   To achieve signing and enveloping, any of the signed-only and
   encrypted-only CMS objects may be nested.

   When nesting is used, backwards compatibility with S/MIME version 2
   requires that each layer of the nested message are identified with
   the OID id-data, and when id-data is used a MIME wrapper is required.
   This can potentially lead to an enormous amount of overhead and
   should be avoided. Because S/MIME version 2 compatibility is of no
   concern, implementations SHOULD directly encode the encapsulated
   object as the eContent of the current structure.

   MIME wrapping to support 7-bit transport is optional and need only be
   used around the outermost CMS structure. In this case, the
   application/pkcs7 content type MUST be used.

   An S/MIME implementation MUST be able to receive and process
   arbitrarily nested CMS structures within reasonable resource limits
   of the recipient computer.

3.4.1. Creating a Triple Wrapped Message With an X.400 Content

   The Enhanced Security Services for S/MIME [ESS] document provides
   examples of how nested, secured S/MIME messages are formatted. ESS
   provides an example of how a triple-wrapped S/MIME message is
   formatted using application/pkcs7-mime for the signatures.

   This section explains how an X.400 content may be conveyed within a
   Triple Wrapped Message because S/MIME version 2 compatibility is of
   no concern:

   Step 1. Start with the X.400 content (called the "original
   content"). The X.400 content MUST be ASN.1 encoded, but SHOULD NOT
   be MIME wrapped.

   Step 2. Place the ASN.1 encoded X.400 content to be protected in the
   SignedData encapContentInfo eContent field. Add any attributes that
   are to be signed.

   Step 3. Sign the result of step 2 (the original content). The
   SignedData encapContentInfo eContentType MUST contain the object
   identifier of the X.400 content.

   Step 4. Encrypt the result of step 3 as a single block. The
   EnvelopedData encryptedContentInfo contentType MUST be set to id-
   signedData. This is called the "encrypted body".

Step 5. Using the same logic as in step 2 and 3 above, sign the
result of step 4 (the encrypted body) as a single block. The
SignedData encapContentInfo eContentType MUST be set to id-
envelopedData. The outer SignedData structure is encapsulated by a
ContentInfo SEQUENCE with a contentType of id-signedData.

Step 6. The resulting message is called the "outer signature", and
is also the triple wrapped message.

MIME wrapping to support 7-bit transport is optional and MUST only be
used around the outermost CMS structure. In this case, the
application/pkcs7-mime content type MUST be used. The smime-type in
the case of adding a MIME wrapper MUST be consistent with that
appropriate to the innermost protection layer.

In some instances, an smime-type will be created that only reflects
one security service (such as certs-only, which applies only to
signed-only messages). However, as new layers are wrapped, this
smime-type SHOULD be propagated upwards. Thus if a certs-only
message were to be encrypted, or wrapped in a new SignedData
structure, the smime-type of certs-only should be propagated up to
the next MIME wrapper. In other words, the innermost type is
reflected outwards.

3.5. Carrying Plaintext X.400 Content in SMTP

While the objectives of this document focus on protecting X.400
content with CMS wrappers, it is a reality that users do not
generally send all message using security. It therefore stands to
reason that a means to carry non-secured X.400 content over the
chosen transport system must be seamlessly provided. While
transporting X.400 content in an X.400 system is trivial, carrying
X.400 content in SMTP requires additional definition.

Content-Type: application/x400-content; content-type = 1*DIGIT *("."
1*DIGIT)

where the content-type parameter value is either a single integer
(for a built-in content-type) or an OID in dotted notation (for an
extended content-type).

4. Use of Certificates

4.1. Certificate Enrollment

   S/MIME v3.1 does not specify how to get a certificate from a
   certificate authority, but instead mandates that every sending agent
   already has a certificate. The PKIX Working Group has, at the time
   of this writing, produced two separate standards for certificate
   enrollment: CMP (RFC 2510) and CMC (RFC 2792).

4.2. Certificate Processing

   A receiving agent MUST provide some certificate retrieval mechanism
   in order to gain access to certificates for recipients of digital
   envelopes. This document does not cover how S/MIME agents handle
   certificates, only what they do after a certificate has been
   validated or rejected. S/MIME certification issues are covered in
   [CERT31].

   At a minimum, for initial S/MIME deployment, a user agent could
   automatically generate a message to an intended recipient requesting
   that recipient's certificate in a signed return message. Receiving
   and sending agents SHOULD also provide a mechanism to allow a user to
   "store and protect" certificates for correspondents in such a way so
   as to guarantee their later retrieval.

4.3. Certificate Name Use for X.400 Content

   End-entity certificates used in the context of this document MAY
   contain an X.400 address as described in [X.400]. The address must
   be in the form of an "ORAddress". The X.400 address SHOULD be in the
   subjectAltName extension, and SHOULD NOT be in the subject
   distinguished name.

   Sending agents SHOULD make the originator address in the X.400
   content (e.g., the "originator" field in P22) match an X.400 address
   in the signer's certificate.

   Receiving agents MUST recognize X.400 addresses in the subjectAltName
   field.

   Receiving agents SHOULD check that the originator address in the
   X.400 content matches an X.400 address in the signer's certificate,
   if X.400 addresses are present in the certificate and an originator
   address is available in the content. A receiving agent SHOULD
   provide some explicit alternate processing of the message if this

comparison fails, which may be to display a message that shows the
recipient the addresses in the certificate or other certificate
details.

The subject alternative name extension is used in S/MIME as the
preferred means to convey the X.400 address(es) that correspond to
the entity for this certificate. Any X.400 addresses present MUST be
encoded using the x400Address CHOICE of the GeneralName type. Since
the SubjectAltName type is a SEQUENCE OF GeneralName, multiple X.400
addresses MAY be present.

5. Security Considerations

This specification introduces no new security concerns to the CMS or
S/MIME models. Security issues are identified in section 5 of [MSG],
section 6 of [ESS] and the Security Considerations section of [CMS].

6. References

6.1. Normative References

[CERT31]   Ramsdell, B., Ed., "Secure/Multipurpose Internet Mail
              Extensions (S/MIME) Version 3.1 Certificate Handling",
              RFC 3850, July 2004.

[CMS]   Housley, R., "Cryptographic Message Syntax (CMS)", RFC
              3852, July 2004.

[CMSAES]   Schaad, J., "Use of the AES Encryption Algorithm in
              CMS", RFC 3565, July 2003.

[CMSALG]   Housley, R., "Cryptographic Message Syntax (CMS)
              Algorithms", RFC 3370, August 2002.

[ESS]   Hoffman, P., Editor "Enhanced Security Services for
              S/MIME", RFC 2634, June 1999.

[MSG]   Ramsdell, B., Ed., "Secure/Multipurpose Internet Mail
              Extensions (S/MIME) Version 3.1 Message Specification",
              RFC 3851, July 2004.

[MUSTSHOULD] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119, March 1997.

[TRANSPORT] Hoffman, P. and C. Bonatti, "Transporting
              Secure/Multipurpose Internet Mail Extensions (S/MIME)
              Objects in X.400", RFC 3855, July 2004.

   [X.400]   ITU-T X.400 Series of Recommendations, Information
                   technology - Message Handling Systems (MHS). X.400:
                   System and Service Overview; X.402: Overall
                   Architecture; X.411: Message Transfer System: Abstract
                   Service Definition and Procedures; X.420: Interpersonal
                   Messaging System; 1996.

6.2. Informative References

   [BODYMAP]   Alvestrand, H., Ed., "Mapping between X.400 and RFC-
                   822/MIME Message Bodies", RFC 2157, January 1998.

   [MIXER]   Kille, S., Ed., "MIXER (Mime Internet X.400 Enhanced
                   Relay): Mapping between X.400 and RFC 822/MIME", RFC
                   2156, January 1998.

   [SMTP]   Klensin, J., "Simple Mail Transfer Protocol", RFC 2821,
                   April, 2001.

7. Editors' Addresses

   Paul Hoffman
   Internet Mail Consortium
   127 Segre Place
   Santa Cruz, CA 95060 USA

   EMail: phoffman@imc.org

   Chris Bonatti
   IECA, Inc.
   15309 Turkey Foot Road
   Darnestown, MD 20878-3640 USA

   EMail: bonattic@ieca.com

   Anders Eggen
   Forsvarets Forskningsinstitutt
   Postboks 25
   2027 Kjeller, Norway

   EMail: anders.eggen@ffi.no

8. **Full Copyright Statement**

   Copyright (C) The Internet Society (2004). This document is subject
   to the rights, licenses and restrictions contained in BCP 78, and
   except as set forth therein, the authors retain all their rights.

   This document and the information contained herein are provided on an
   "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS
   OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET
   ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED,
   INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
   INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
   WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

   The IETF takes no position regarding the validity or scope of any
   Intellectual Property Rights or other rights that might be claimed to
   pertain to the implementation or use of the technology described in
   this document or the extent to which any license under such rights
   might or might not be available; nor does it represent that it has
   made any independent effort to identify any such rights. Information
   on the procedures with respect to rights in RFC documents can be
   found in BCP 78 and BCP 79.

   Copies of IPR disclosures made to the IETF Secretariat and any
   assurances of licenses to be made available, or the result of an
   attempt made to obtain a general license or permission for the use of
   such proprietary rights by implementers or users of this
   specification can be obtained from the IETF on-line IPR repository at
   http://www.ietf.org/ipr.

   The IETF invites any interested party to bring to its attention any
   copyrights, patents or patent applications, or other proprietary
   rights that may cover technology that may be required to implement
   this standard. Please address the information to the IETF at ietf-
   ipr@ietf.org.

Acknowledgement

   Funding for the RFC Editor function is currently provided by the
   Internet Society.