Abstract

In the current S/MIME Version 3.1 specification, the header protection is achieved by encoding the whole message as a message/rfc822 MIME object. Since this approach poses some practical problems, we propose to use signed attributes to implement a fully backward compatible S/MIME header protection scheme.
Table of Contents

1. Introduction ............................................. 3
   1.1. Terminology ........................................ 3
   1.2. Syntactic Notation ................................. 3
   1.3. Object Identifiers ................................. 4
   1.4. Security Goals of Header Protection ............... 4
   1.5. Header Protection in S/MIME Version 3.1 .......... 4
   1.6. Prototype Implementation ......................... 5
2. S/MIME Header Protection Entity .......................... 5
   2.1. Fieldname List .................................... 5
   2.2. Canonicalization of Headers ....................... 7
3. CMS Fields ............................................. 8
   3.1. CanonAlgorithmIdentifier ......................... 8
   3.2. SMIME Header Protection ......................... 9
4. Creating Signed S/MIME Messages with Header Protection ... 9
   4.1. Preparing an SMIME-Header-Protection Attribute ... 9
5. Verifying Signed S/MIME Message with Header Protection ... 10
   5.1. Verifying an SMIME-Header-Protection Attribute ... 10
6. Security Considerations .................................. 10
7. References ........................................... 11
   7.1 Normative References ................................ 11
   7.2 Informative References ............................... 11
A. ASN.1 Module .......................................... 11
B. Examples ............................................... 12
   B.1. SMIME-Header-Protection Attribute with "simple"
       and "SHA256" ......................................... 12
   B.2. SMIME-Header-Protection Attribute with "relaxed"
       and "SHA1" ........................................... 13
C. Authors’ Addresses ..................................... 14
Full Copyright Statement .................................. 14
Intellectual Property ..................................... 15
Acknowledgment .......................................... 15
1. Introduction

Mail message header fields as defined in [RFC2822] contain security critical information that is not protected cryptographically. The only exception is the header fields From and Sender. Receiving agents MUST check that the address in the header field From or Sender of a mail message matches an Internet mail address, if present, in the signer's certificate, if mail addresses are present in the certificate. 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 [RFC3850]. Other header fields like "To", "Date", "Reply-To" and "Subject" remain totally unprotected.

In the solution described in this specification, a digest value is computed over the canonicalized version of some selected header fields. This technique resembles header protection in [RFC4871]. Then the digest value is included in a signed attribute field of a CMS signature.

This solution allows conforming clients to check if one of the selected header fields has been altered by simply re-computing the digest value. Non-conforming legacy clients will simply ignore that the signed attribute contains a digest value, and will only check the digest value computed over the message body according to S/MIME.

By including separate signed attributes for different header fields, each field can be protected separately.

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 [RFC2119].

1.2. Syntactic Notation

The following tokens are imported from other RFCs as noted. Those RFCs should be considered definitive.

The following tokens are imported from [RFC2822]:

  o "field-name" (name of a header field)

Other tokens not defined herein are imported from [RFC4234]. These are intuitive primitives such as SP, HTAB, WSP, ALPHA, DIGIT, CRLF, etc.
1.3. Object Identifiers

The object identifiers defined in this specification is only for the experiment. When this memo moves to standards track within the IETF, it is intended that the IANA will maintain this registry.

1.4. Security Goals of Header Protection

The main security goal of mail message header protection is not to protect the whole RFC 2822 header against manipulation, but to make it possible for the receiving client to detect which headers have been changed.

1.5. Header Protection in S/MIME Version 3.1

S/MIME Version 3.1 [RFC3851] addresses the header protection by including all header fields as generated by the sending mail client, together with the body of the message, in a message/rfc822 mime object, which can then be protected by S/MIME. It is up to the receiving client to decide how to present this message to the user.

This approach has, however, some limitations: If some of the message headers are changed during transport (e.g. when sent to a mailing list), this will either invalidate the whole message, or not be detected at all, depending on the receiving mail client’s behavior.

This approach has the following disadvantages:

- All inner header fields must also appear in the outer header (i.e., those headers must be presented doubly) so that the mail message is conform to [RFC2822] and the mail server and relay systems know how to send the mail message.

- Only the inner header fields are protected, but not the outer header fields. As stated in [RFC3851], it is up to the receiving client to decide how to present the inner header along with the unprotected outer header. Usually the following header fields, if present, are shown in most clients: "From", "Sender", "To", "CC", "Date", and "Subject". If the same header field is present in both inner and outer header, only the one in the inner header is presented. If a header field is only presented in the outer header, it will be also shown. Most mail messages do not contain the headers "Sender" and "CC", hence one can add these header fields in the outer header to confuse the receivers.

- It complicates the receiver to show the mail message. It is difficult to determine whether the message within the message/rfc822 wrapper is the top-level message or the complete message/rfc822 MIME entity is another encapsulated mail message.
1.6. Prototype Implementation

A prototype implementation of this memo is available in [FeLi08]. When this memo moves to standards track within the IETF, this section will be removed.

2. S/MIME Header Protection Entity

A smime header protection entity contains names of header fields to be protected, the canonicalization algorithm, the digest algorithm and the corresponding digest value.

2.1. Fieldname List

The fieldname-list is a colon-separated list of header field names that identify the header fields presented to the digest algorithm; it is defined as follows:

```
fieldname-list  = field-name *("":"" field-name)
```

The fieldname-list MUST contain the complete list of header fields in the order presented to the digest algorithm. The field MAY contain names of header fields that do not exist when digested; nonexistent header fields do not contribute to the digest value computation (that is, they are treated as the null input, including the header field name, the separating colon, the header field value, and any CRLF terminator).

The fieldname-list is compared against actual header field names in a case insensitive manner.

Signers choosing to protect an existing header field that occurs more than once in the message (such as "Resent-From") MUST protect the physically last instance of that header field in the header block. Signers wishing to protect multiple instances of such a header field MUST include the header field name multiple times and MUST protect such header fields in order from the bottom of the header field block to the top. The signer MAY include more instances of a header field name than there are actual corresponding header fields to indicate that additional header fields of that name SHOULD NOT be added.
INFORMATIVE EXAMPLE:

Given a mail message as follows:

Resent-From: A <CRLF>
Resent-From: B <CRLF>
From: C <CRLF>
To: D <CRLF>
CC: E <CRLF>
Subject: F <CRLF>

Body

If the signer wishes to sign the header fields "To", "CC", "From", and "Subject" in that order, then the fieldname-list is:

To:CC:From:Subject

and the following header fields will be digested in the order:

To: D <CRLF>
CC: E <CRLF>
From: C <CRLF>
Subject: F <CRLF>

If the signer wishes to protect the header fields "To", "CC", "From", "Subject", and the "Resent-From" header field with the value B, then the fieldname-list is:

To:CC:From:Subject:Resent-From

and the following header fields will be digested in the order:

To: D <CRLF>
CC: E <CRLF>
From: C <CRLF>
Subject: F <CRLF>
Resent-From: B <CRLF>

If the signer wishes to protect the header fields "To", "CC", "From", "Subject", and both "Resent-From" header fields, and wishes to prevent from adding new "Resent-From" header field, then the fieldname-list is:

To:CC:From:Subject:Resent-From:Resent-From:Resent-From
and the following header fields will be digested in the order:

To: D  
CC: E  
From: C  
Subject: F  
Resent-From: B  
Resent-From: A

If the receiving agents receives the message with a third
Resent-From: G that is inserted before Resent-From: A, then the
following header fields will be used to verify S/MIME header
protection entity:

To: D  
CC: E  
From: C  
Subject: F  
Resent-From: B  
Resent-From: A  
Resent-From: G

then the verification fails.

Signers should be careful of digesting header fields that might have
additional instances added later in the delivery process, since such
header fields might be inserted after the digested instance or
otherwise reordered. Trace header fields (such as "Received") and
Resent-* blocks are the only fields prohibited by [RFC2822] from
being reordered.

INFORMATIVE ADMONITION: Despite the fact that [RFC2822] permits
header fields to be reordered (with the exception of Received
header fields), reordering of digested header fields with multiple
instances by intermediate MTAs will cause S/MIME header protection
entity in this specification to be broken; such anti-social
behavior should be avoided.

INFORMATIVE IMPLEMENTER’S NOTE: Although not required by this
specification, all end-user visible header fields should be
digested to avoid possible "indirect spamming". For example, if
the "Subject" header field is not digested, a spammer can resend
a previously digested mail, replacing the legitimate subject with
a one-line spam.

2.2. Canonicalization of Headers

Mail message, specially the mail message header, may be modified by
some mail servers and relay systems. Some signers may demand that any
modification of the mail message header result in a signature failure, while some other signers may accept modification of the header within the bounds of mail message standards such as [RFC2822].

To satisfy all requirements, two canonicalization algorithms are defined for each of the header and the body: a "simple" algorithm stated in Section 3.4.1 of [RFC4871] that tolerates almost no modification and a "relaxed" algorithm stated in Section 3.4.2 of [RFC4871] that tolerates common modifications such as white-space replacement and header field line re-wrapping.

3. CMS Fields

3.1. CanonAlgorithmIdentifier

The CanonAlgorithmIdentifier type identifies a canonicalization algorithm. Examples include "simple" header canonicalization, and "relaxed" header canonicalization.

CanonAlgorithmIdentifier ::= AlgorithmIdentifier

AlgorithmIdentifier is defined in [RFC3280] as follows:

AlgorithmIdentifier ::= SEQUENCE {
  algorithm               OBJECT IDENTIFIER,
  parameters              ANY DEFINED BY algorithm OPTIONAL  }

The algorithm identifier is used to identify a canonicalization algorithm.

The "simple" canonicalization algorithm is identified by the following object:

id-alg-simpleHeaderCanon OBJECT IDENTIFIER ::= {iso(1)
  member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
  smime(16) alg(3) 101}

The "relaxed" canonicalization algorithm is identified by the following object:

id-alg-relaxedHeaderCanon OBJECT IDENTIFIER ::= {iso(1)
  member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
  smime(16) alg(3) 102}

For the canonicalization algorithms "simple" and "relaxed" the parameters field is NULL.
3.2. SMIME Header Protection

The smime-header-protection attribute type specifies the S/MIME header protection entity. It MUST be a signed attribute or an authenticated attribute; it MUST NOT be an unsigned attribute, unauthenticated attribute, or unprotected attribute in CMS signature.

The following object identifier identifies the smime-header-protection attribute:

```
id-smimeHeaderProtection OBJECT IDENTIFIER ::= {iso(1)
    member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) aa(2) 101}
```

The attrValues of the smime-header-protection attribute contains only one value that has ASN.1 type SMIMEHeaderProtectionEntity:

```
SMIMEHeaderProtectionEntity ::= SEQUENCE {
    canonAlgorithm       CanonAlgorithmIdentifier,
    digestAlgorithm      DigestAlgorithmIdentifier,
    headerfieldNames     PrintableString,
    digest               Digest
}
```

The canonAlgorithm field specifies the canonicalization algorithm. The digestAlgorithm field specifies the digest algorithm. The format of an headerfieldNames is a "headername-list" field specified in Section 2.1. The headerfieldNames field specifies the list of header field names. The digest field carries the the digest value.

4. Creating Signed S/MIME Messages with Header Protection

The signed S/MIME messages with header protection are created same as in [RFC3851] except the followings:

- Before computing the digest value over the signedAttrs, the smime-header-protection attribute MUST be prepared (see Section 4.1) and added to the signedAttrs.
- All header fields that are protected MUST be prepared before the preparing the smime-header-protection.

4.1. Preparing an SMIME-Header-Protection Attribute

An smime-header-protection attribute is prepared as follows:

Step 1. Choose the canonicalization algorithm, the digest algorithm, and the list of names of message header fields to be digested. The digest algorithm SHOULD be the same as the digest algorithm in the
SignerInfo which the smime-header-protection attribute should be added to.

Step 2. Retrieve the message header fields from the message according to the protected header fields from Step 1.

Step 3. Canonicalize the retrieved header fields from Step 2 according to the canonicalization algorithm.

Step 4. Compute the digest value over the canonicalization result in Step 3 according to the digest algorithm.

Step 5. Create an smime-header-protection attribute. Store the chosen canonicalization algorithm, the digest algorithm, and the list of names from Step 1 in ASN.1 fields canonAlgorithm, digestAlgorithm, and headerfieldNames, respectively. Store the digest value from Step 4 in the ASN.1 field digest.

5. Verifying Signed S/MIME Message with Header Protection

The signed S/MIME message with header protection are verified first same as in [RFC3851], then the smime-header-protection attribute is verified as stated in Section 5.1.

5.1. Verifying an SMIME-Header-Protection Attribute

An smime-header-protection attribute is verified as follows:

Step 1. Retrieve the canonicalization algorithm, the digest algorithm, and the list of names of message header fields, and the digest value from the smime-header-protection attribute.

Step 2. Retrieve the message header fields from the message according to the protected header fields from Step 1.

Step 3. Canonicalize the retrieved header fields from Step 2 according to the canonicalization algorithm.

Step 4. Compute the digest value over the canonicalization result in Step 3 according to the digest algorithm.

Step 5. Compares the computed digest value from Step 4 and the stored one from Step 1. If both digest values are different, then the verification fails; otherwise the verification successes.

6. Security Considerations

All security considerations from [RFC3851] and [RFC3852] apply to applications that use procedures described in this document.
7. References

7.1 Normative References


7.2 Informative References


A. ASN.1 Module

SMIMEHeaderProtectionService
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)  pkcs-9(9) smime(16) modules(0) shps(101) }

DEFINITIONS IMPLICIT TAGS ::= BEGIN IMPORTS

Liao & Schwenk Expires October 19, 2008
-- Imports from RFC 3280
AlgorithmIdentifier
FROM PKIX1Explicit88
  { iso(1) identified-organization(3) dod(6)
    internet(1) security(5) mechanisms(5) pkix(7)
    mod(0) pkix1-explicit(18) }

-- Imports from RFC 3852
DigestAlgorithmIdentifier, Digest
FROM CryptographicMessageSyntax2004
  { iso(1) member-body(2) us(840) rsadsi(113549)
    pkcs(1) pkcs-9(9) smime(16) modules(0) cms-2004(24) }

CanonAlgorithmIdentifier ::= AlgorithmIdentifier

id-alg-simpleHeaderCanon OBJECT IDENTIFIER ::= {iso(1)
  member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
  smime(16) alg(3) 101}

id-alg-relaxedHeaderCanon OBJECT IDENTIFIER ::= {iso(1)
  member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
  smime(16) alg(3) 102}

id-smimeHeaderProtection OBJECT IDENTIFIER ::= {iso(1)
  member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
  smime(16) aa(2) 101}

SMIMEHeaderProtectionEntity ::= SEQUENCE {
  canonAlgorithm     CanonAlgorithmIdentifier,
  digestAlgorithm    DigestAlgorithmIdentifier,
  headerfieldNames   PrintableString,
    -- The format of a headerfieldNames
    -- is a "headername-list" field
    -- specified in Section 2.1.
  digest            Digest
}

END

B. Examples

B.1. SMIME-Header-Protection Attribute with "simple" and "SHA256"

This section contains an annotated hex dump of a 115 byte
smime-header-protection attribute which is contained in the
signedAttrs of a signature. The attribute contains the following
information:
(a) the canocalization algorithm is "simple" header canonicalization;
(b) the digest algorithm is "SHA256";
(c) the list of header field names is "From:Sender:To:Cc:Date:Subject";
(d) the digest value (32 hex)

0 30  115: SEQUENCE {
  2 06   11:  OBJECT IDENTIFIER
        :    smime-header-protection {1 2 840 113549 1 9 16 2
        :       101}
  15 31  100:  SET {
    17 30  98:     SEQUENCE {
      19 30  15:       SEQUENCE {
        21 06   11:         OBJECT IDENTIFIER
          :           simple { 1 2 840 113549 1 9 16 3 101 }
        34 05    0:         NULL
        :       }
      36 30  13:       SEQUENCE {
        38 06    9:         OBJECT IDENTIFIER
          :           SHA256 { 2 16 840 1 101 3 4 2 1 }
        49 05    0:         NULL
        :       }
      51 16   30:       PrintableString "From:Sender:To:Cc:Date:Subject"
        :             6B 79 C7 93 F1 87 89 A1 11 66 A8 10 83 42 24 53 AB
        :             BA F1 D4 FD 95 EB 8B FA 55 F6 31 52 E7 86 50
        :           }
      :       }
  :     }

B.2. SMIME-Header-Protection Attribute with "relaxed" and "SHA1"

This section contains an annotated hex dump of a 100 byte
smime-header-protection attribute which is contained in the
signedAttrs of a signature. The attribute contains the following
information:

(a) the canocalization algorithm is "relaxed" header
canonicalization;
(b) the digest algorithm is "SHA1";
(c) the list of header field names is
"From:Sender:To:Cc:Date:Subject";
(d) the digest value (20 hex)

0 30  100: SEQUENCE {
  2 06   11:  OBJECT IDENTIFIER
        :    smime-header-protection {1 2 840 113549 1 9 16 2
        :       101}
C. Authors’ Addresses

Lijun Liao
Chair for Network and Data Security
Ruhr-University Bochum
44780 Bochum
Germany
Mail message: lijun.liao@nds.rub.de

Joerg Schwenk
Chair for Network and Data Security
Ruhr-University Bochum
44780 Bochum
Germany
Mail message: joerg.schwenk@nds.rub.de

Full Copyright Statement

Copyright (C) The IETF Trust (2008).

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, THE IETF TRUST AND
THE INTERNET ENGINEERING TASK FORCE DISCLAIMS 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

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.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF
Administrative Support Activity (IASA).