Independent Submission
Request for Comments: 7508
Category: Experimental
ISSN: 2070-1721

L. Cailleux
DGA MI
C. Bonatti
IECA

April 2015

Securing Header Fields with S/MIME

Abstract

This document describes how the S/MIME protocol can be extended in order to secure message header fields defined in RFC 5322. This technology provides security services such as data integrity, non-repudiation, and confidentiality. This extension is referred to as 'Secure Headers'.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7508.

Table of Contents

1. Introduction

The S/MIME [RFC5751] standard defines a data encapsulation format for the achievement of end-to-end security services such as integrity, authentication, non-repudiation, and confidentiality. By default, S/MIME secures message body parts, at the exclusion of the message header fields.

S/MIME provides an alternative solution to secure header fields: "the sending client MAY wrap a full MIME message in a message/rfc822 wrapper in order to apply S/MIME security services to header fields". However, the S/MIME solution doesn't provide any guidance regarding what subset of message header fields to secure, procedures for clients to reconcile the "inner" and "outer" headers, or procedures for client interpretation or display of any failures.

Several other security specifications supplement S/MIME features but fail to address the target requirement set of this document. Such other security specifications include DomainKeys Identified Mail (DKIM) [RFC6376], STARTTLS [RFC3207], TLS with IMAP [RFC2595], and an

Internet-Draft referred to as "Protected Headers" [PRHDRS]. An explanation of what these services accomplish and why they do not solve this problem can be found in subsequent sections.

The goal of this document is to define end-to-end secure header field mechanisms compliant with S/MIME standard. This technique is based on the signed attribute fields of a Cryptographic Message Syntax (CMS) [RFC5652] signature.

2. Terminology and Conventions Used in This Document

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].

The terms Message User Agent (MUA), Message Submission Agent (MSA), and Message Transfer Agent (MTA) are defined in the email architecture document [RFC5598].

The term Domain Confidentiality Authority (DCA) is defined in the S/MIME Domain Security specification [RFC3183].

End-to-end Internet Mail exchanges are performed between message originators and recipients.

The term message header fields is described in [RFC5322]. A header field is composed of a name and a value.

Secure Headers technology uses header field statuses required to provide a confidentiality service toward message headers. The following three terms are used to describe the field statuses:

- duplicated (the default status). When this status is present or if no status is specified, the signature process embeds the header field value in the digital signature, but the value is also present in the message header fields.

- deleted. When this status is present, the signature process embeds the header field value in the digital signature, and the encryption process deletes this field from the message to preserve its confidentiality.

- modified. When this status is present, the signature process embeds the header field value in the digital signature, and the encryption process modifies the value of the header field in the message. This preserves confidentiality and informs a receiver’s

> noncompliant MUA that secure headers are being used. New values for each field might be configured by the sender (i.e., "This header is secured; use a compliant client.").

The term non-repudiation is used throughout this document in deference to the usage in the S/MIME Message Specification [RFC5751]. It is recognized that this term carries with it much baggage, and that there is some disagreement as to its proper meaning and usage. However, in the context of this document, the term merely refers to one set of possible security services that a conforming implementation might be able to provide. This document specifies no normative requirements for non-repudiation.

3. Context

Over the Internet, email use has grown and today represents a fundamental service. Meanwhile, continually increasing threat levels are motivating the implementation of security services.

Historically, SMTP [RFC5321] and the Internet Message Format (IMF) [RFC5322] don’t provide, by default, security services. The S/MIME standard [RFC5751] was published in order to address these needs. S/MIME defines a data encapsulation format for the provision of end-to-end security services such as integrity, authentication, non-repudiation, and confidentiality. By default, S/MIME secures message body parts, at the exclusion of the message header fields. In order to protect message header fields (for instance, the "Subject", "To", "From", or customized fields), several solutions exist.

In Section 3.1 of [RFC5751], S/MIME defines an encapsulation mechanism:

> [...] the sending client MAY wrap a full MIME message in a message/rfc822 wrapper in order to apply S/MIME security services to these header fields. It is up to the receiving client to decide how to present this "inner" header along with the unprotected "outer" header.

However, some use cases are not addressed, especially in the case of message encryption. What happens when header fields are encrypted? How does the receiving client display these header fields? How can a subset of header fields be secured? S/MIME doesn’t address these issues.

Some partial header protection is provided by the S/MIME Certificate Handling specification [RFC5750]:

> Receiving agents MUST check that the address in the From or Sender header of a mail message matches an Internet mail address, if present, in the signer's certificate, if mail addresses are present in the certificate.

In some cases, this may provide assurance of the integrity of the From or Sender header values. However, the solution in RFC 5750 only provides a matching mechanism between email addresses and provides no protection to other header fields.

Other security specifications (introduced below) exist such as DKIM, STARTTLS and TLS with IMAP, but they meet other needs (signing domain, secure channels, etc.).

STARTTLS and TLS with IMAP provide secure channels between components of the email system (MUA, MSA, MTA, etc.), but end-to-end integrity cannot be guaranteed.

DKIM defines a domain-level authentication framework for email. While this permits integrity and origination checks on message header fields and the message body, it does this for a domain actor (usually the SMTP service or equivalent) and not for the entity that is sending, and thus signing, the message. (Extensions to DKIM might be able to solve this issue by authenticating the sender and making a statement of this fact as part of the signed message headers.) DKIM is also deficient for our purposes, as it does not provide a confidentially service.

An Internet-Draft referred to as "Protected Headers" [PRHDRS] has been proposed. Mechanisms described in that document are the following:

> [...] 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.

(Note that RFC 4871 has been obsoleted by RFC 6376.)

That specification doesn't address all conceivable requirements as noted below. If the protected header field has been altered, the original value cannot be determined by the recipient. In addition, the encryption service cannot provide confidentiality for fields that must remain present in the message header during transport.

This document proposes a technology for securing message header fields. It's referred to as "Secure Headers". It is based on S/MIME and CMS standards. It provides security services such as data integrity, confidentiality, and non-repudiation of the sender. Secure Headers is backward compatible with other S/MIME clients. S/MIME clients who have not implemented Secure Headers technology need merely ignore specific signed attributes fields in a CMS signature (which is the default behavior).

4. Mechanisms to Secure Message Header Fields

Secure Headers technology involves the description of a security policy. This policy MUST describe a secure message profile and list the header fields to secure. How this security policy is agreed upon or communicated is beyond the scope of this document.

Secure headers are based on the signed attributes field as defined in CMS. The details are as follows. The message header fields to be secured are integrated in a structure (SecureHeaderFields structure) that is encapsulated in the signed attributes structure of the SignerInfo object. There is only one value of HeaderFields encoded into a single SignedAttribute in a signature. See Appendix A for an example. For each header field present in the secure signature, a status can be set. Then, as described in Section 5.4 of CMS [RFC5652], the message digest calculation process computes a message digest on the content together with the signed attributes. Details of the signature generation process are in Section 4.5.1 of this document.

Verification of secure header fields is based on the signature verification process described in CMS. At the end of this process, a comparison between the secure header fields and the corresponding message header fields is performed. If they match, the signature is valid. Otherwise, the signature is invalid. Details of the signature verification process are in Section 4.5.2 of this document.

Non-conforming S/MIME clients will ignore the signed attribute containing the SecureHeaderFields structure, and only perform the verification process described in CMS. This guarantees backward compatibility.

Secure headers provide security services such as data integrity, non-repudiation, and confidentiality.

For different reasons (e.g., usability, limits of IMAP [RFC3501]), encryption and decryption processes are performed by a third party. The third party that performs these processes is referred to in the Domain Security specification as a Domain Confidentiality Authority (DCA). Details of the encryption and decryption processes are in Sections 4.6.1 and 4.6.2 of this document.

The architecture of Secure Headers is presented below. The MUA performs the signature generation process (C) and signature verification process (F). The DCA performs the message encryption process (D) and message decryption process (E). The encryption and decryption processes are optional.

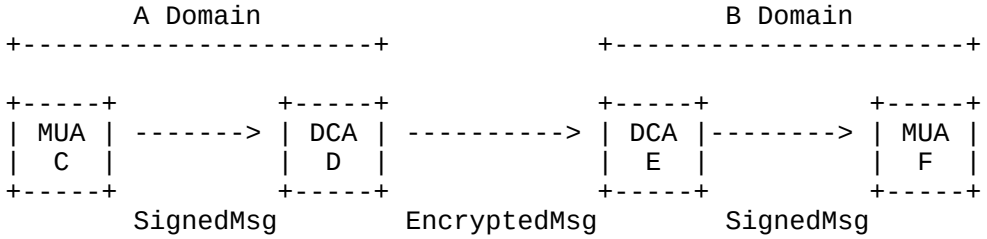

Figure 1: Architecture of Secure Headers

4.1. ASN.1 Syntax of Secure Header Fields

The ASN.1 syntax [ASN1-88] of the SecureHeaderFields structure is as follows:

```
SecureHeaderFields ::= SET {
   canonAlgorithm Algorithm,
   secHeaderFields HeaderFields }

id-aa-secureHeaderFieldsIdentifier OBJECT IDENTIFIER ::= {
   iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
   pkcs-9(9) smime(16) id-aa(2) secureHeaderFieldsIdentifier(55) }

Algorithm ::= ENUMERATED {
   canonAlgorithmSimple(0),
   canonAlgorithmRelaxed(1) }

HeaderFields ::= SEQUENCE SIZE (1..MAX) OF HeaderField

HeaderField ::= SEQUENCE {
   field-Name HeaderFieldName,
   field-Value HeaderFieldValue,
   field-Status HeaderFieldStatus DEFAULT duplicated }
```

```
HeaderFieldName ::= VisibleString (FROM (ALL EXCEPT (":")))
     -- This description matches the description of
     -- field name in Sections 2.2 and 3.6.8 of RFC 5322

HeaderFieldValue ::= UTF8String
     -- This description matches the description of
     -- field body in Section 2.2 of RFC 5322 as
     -- extended by Section 3.1 of RFC 6532.

HeaderFieldStatus ::= INTEGER {
   duplicated(0), deleted(1), modified(2) }
```

4.2. Secure Header Fields Length and Format

This specification requires MUA security capabilities in order to process well-formed headers, as specified in IMF. Notice that it includes long header fields and folded header fields.

4.3. Canonicalization Algorithm

During a message transfer through a messaging system, some components might modify headers (i.e., adding or deleting space, changing or lowercase or uppercase). This might lead to a comparison mismatch of header fields. This emphasizes the need of a conversion process in order to transform data to their canonical form. This process is named the canonicalization process.

Two canonicalization algorithms are considered here, according to Section 3.4 of the DKIM specification [RFC6376]. The "simple" algorithm doesn’t allow any modification, whereas the "relaxed" algorithm accepts slight modifications like space replacement or line reformatting. Given the scope of this document, canonicalization mechanisms only involve header fields.

Implementations SHOULD use the "relaxed" algorithm to promote interoperability with non-conforming SMTP products.

4.4. Header Field Statuses

Header field statuses are necessary to provide a confidentiality service for message headers. In this specification, the confidentiality of header fields is provided by the DCA. This point is described in Section 4. The DCA performs the message encryption process and message decryption process; these processes are described in detail in Sections 4.6.1 and 4.6.2. Although header field statuses are embedded in the signature, the signature processes

(generation and verification) ignore them. The header field status defaults to "duplicated". If the header field is confidential, the header field status MUST be either "deleted" or "modified".

4.5. Signature Process

4.5.1. Signature Generation Process

During the signature generation process, the sender's MUA MUST embed the SecureHeaderFields structure in the signed attributes, as described in CMS. The SecureHeaderFields structure MUST include a canonicalization algorithm.

The sender's MUA MUST have a list of header fields to secure, statuses, and a canonicalization algorithm, as defined by the security policy.

Header fields (names and values) embedded in signed attributes MUST be the same as those included in the initial message.

If different headers share the same name, all instances MUST be included in the SecureHeaderFields structure.

If multiple signatures are used, as explained in the CMS and Multiple Signer [RFC4853] specifications, the SecureHeaderFields structure MUST be the same in each SignerInfos object.

If a header field is present and its value is empty, HeaderFieldValue MUST have a zero-length field-Value.

Considering secure header mechanisms, the signature generation process MUST perform the following steps:

1) Select the relevant header fields to secure. This subset of headers is defined according the security policy.

2) Apply the canonicalization algorithm for each selected header field.

3) Complete the following fields in the SecureHeaderFields structure according to the initial message: HeaderFieldName, HeaderFieldValue, and HeaderFieldStatus.

4) Complete the algorithm field according to the canonicalization algorithm configured.

5) Embed the SecureHeaderFields structure in the signed attributes of the SignerInfos object.

6) Compute the signature generation process as described in Section 5.5 of CMS [RFC5652].

4.5.2. Signature Verification Process

During the signature verification process, the receiver's MUA compares header fields embedded in the SecureHeaderFields structure with those present in the message. For this purpose, it uses the canonicalization algorithm identified in the signed attributes. If a mismatch appears during the comparison process, the receiver's MUA MUST invalidate the signature. The MUA MUST display information on the validity of each header field. It MUST also display the values embedded in the signature.

The receiver's MUA MUST know the list of mandatory header fields in order to verify their presence in the message. If a header field defined in a message is in the secure header list, it MUST be included in the SecureHeaderFields structure. Otherwise, the receiver's MUA MUST warn the user that a non-secure header is present.

Considering secure header mechanisms, the signature verification process MUST perform the following steps:

1) Execute the signature verification process as described Section 5.6 of CMS [RFC5652]. If the signature appears to be invalid, the process ends. Otherwise, the process continues.

2) Read the type of canonicalization algorithm specified in the SecureHeaderFields structure.

3) For each field present in the signature, find the matching header in the message. If there is no matching header, the verification process MUST warn the user, specifying the missing header name. The signature is tagged as invalid. Note that any header fields encrypted as per Section 4.6 (i.e., status of "deleted" or "modified") have been are already restored by the DCA when the signature verification process is performed by the MUA.

4) Compute the canonicalization algorithm for each header field value in the message. If the "simple" algorithm is used, the steps described in Section 3.4.1 of DKIM [RFC6376] are performed. If the relaxed algorithm is used, the steps described in Section 3.4.2 of DKIM [RFC6376] are performed.

5) For each field, compare the value stored in the SecureHeaderFields structure with the value returned by the canonicalization algorithm. If the values don't match, the verification process MUST warn the user. This warning MUST mention mismatching fields. The signature is tagged as invalid. If all the comparisons succeed, the verification process MUST also notify the user (i.e., using an appropriate icon).

6) Verify that no secure header has been added to the message header, given the initial fields. If an extra header field has been added, the verification process MUST warn the user. This warning MUST mention extra fields. The signature is tagged as invalid. This step is only performed if the sender and the recipient share the same security policy.

7) Verify that each mandatory header in the security policy and present in the message is also embedded in the SecureHeaderFields structure. If such headers are missing, the verification process MUST warn the user and indicate the names of the missing headers.

The MUA MUST display the properties of each secure header field (name, value, and status) and the canonicalization algorithm used.

4.6. Encryption and Decryption Processes

Encryption and decryption operations are not performed by MUAs. This is mainly justified by limitations of existing email delivery protocols, for example, IMAP. The solution developed here relies on concepts explained in Section 4 of the Domain Security specification [RFC3183]. A fundamental component of the architecture is the Domain Confidentiality Authority (DCA). Its purpose is to encrypt and decrypt messages instead of that being performed by senders and receivers (respectively).

4.6.1. Encryption Process

All the computations presented in this section MUST be performed only if the following conditions are verified:

- The content to be encrypted MUST consist of a signed message (application/pkcs7-mime with SignedData, or multipart/signed) as shown in Section 3.4 of the S/MIME specification [RFC5751].

- A SecureHeaderFields structure MUST be included in the signedAttrs field of the SignerInfo object of the signature.

All the mechanisms described below MUST start at the beginning of the encryption process, as explained in CMS. They are performed by the sender's DCA. For extraction of the field status, the following steps MUST be performed for each field included in the SecureHeaderFields structure:

1. If the status is "duplicated", the field is left at its existing value.

2. If the status is "deleted", the header field (name and value) is removed from the message. Mandatory header fields specified in [RFC5322] MUST be kept.

3. If the status is "modified", the header value is replaced by a new value, as configured in the DCA.

4.6.2. Decryption Process

All the computations presented in this section MUST be performed only if the following conditions are verified:

- The decrypted content MUST consist of a signature object or a multipart object, where one part is a detached signature, as shown in Section 3.4 of the S/MIME specification [RFC5751].

- A SecureHeaderFields structure MUST be included in the SignerInfo object of the signature.

All the mechanisms described below MUST start at the end of the decryption process, as explained in CMS. They are executed by the receiver's DCA. The following steps MUST be performed for each field included in the SecureHeaderFields structure:

1. If the status is "duplicated", the field is left at its existing value.

2. If the status is "deleted", the DCA MUST write a header field (name and value) in the message. This header MUST be compliant with the information embedded in the signature.

3. If the status is "modified", the DCA MUST rewrite a header field in the message. This header MUST be compliant with the SecureHeaderFields structure.

5. Case of Triple Wrapping

Secure Headers mechanisms MAY be used with triple wrapping, as described in Enhanced Security Services (ESS) [RFC2634]. In this case, a SecureHeaderFields structure MAY be present in the inner signature, the outer signature, or both. In the last case, the two SecureHeaderFields structures MAY differ. One MAY consider the encapsulation of a header field in the inner signature in order to satisfy confidentiality needs. On the contrary, an outer signature encapsulation MAY help for delivery purposes. The sender's MUA and receiver's MUA must have a security policy for triple wrapping. This security policy MUST be composed of two parts -- one for the inner signature and the other for the outer signature.

6. Security Gateways

Some security gateways sign or verify messages that pass through them. Compliant gateways MUST apply the process described in Section 4.5.

For noncompliant gateways, the presence of a SecureHeaderFields structure does not change their behavior.

In some case, gateways MUST generate a new signature or insert signerInfos into the signedData block. The format of signatures generated by gateways is outside the scope of this document.

7. Security Considerations

This specification describes an extension of the S/MIME standard. It provides message header integrity, non-repudiation, and confidentiality. The signature and encryption processes are complementary. However, according to the security policy, only the signature mechanism is applicable. In this case, the signature process is implemented between MUAs. The encryption process requires signed messages with the Secure Headers extension. If required, the encryption process is implemented by DCAs.

This specification doesn't address end-to-end confidentiality for message header fields. Messages sent and received by MUAs could be transmitted as plaintext. In order to avoid interception, the use of TLS is recommended between MUAs and DCAs (uplink and downlink). Another solution might be the use of S/MIME between MUAs and DCAs in the same domain.

For the header field confidentiality mechanism to be effective, all DCAs supporting confidentiality must support Secure Headers processing. Otherwise, there is a risk that headers are not obscured

upon encryption or not restored upon decryption. In the former case, confidentiality of the header fields is compromised. In the latter case, the integrity of the headers will appear to be compromised.

8. IANA Considerations

IANA has registered value 65, mod-sMimeSecureHeadersV1, in the "SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)" registry.

IANA has also registered value 55, id-aa-secureHeaderFieldsIdentifier, in the "SMI Security for S/MIME Attributes (1.2.840.113549.1.9.16.2)" registry. This value will be used to identify an authenticated attribute carried within a CMS wrapper [RFC5652]. This attribute OID appears in Section 4.1 and again in the reference definition in Appendix A.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2634] Hoffman, P., Ed., "Enhanced Security Services for S/MIME", RFC 2634, June 1999, <http://www.rfc-editor.org/info/rfc2634>.

[RFC4853] Housley, R., "Cryptographic Message Syntax (CMS) Multiple Signer Clarification", RFC 4853, April 2007, <http://www.rfc-editor.org/info/rfc4853>.

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

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

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

[ASN1-88] CCITT, Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1), 1988.

9.2. Informative References

[PRHDRS] Liao, L. and J. Schwenk, "Header Protection for S/MIME", draft-liao-smimeheaderprotect-05, Work in Progress, June 2009.

[RFC2595] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, June 1999, <http://www.rfc-editor.org/info/rfc2595>.

[RFC3183] Dean, T. and W. Ottaway, "Domain Security Services using S/MIME", RFC 3183, October 2001, <http://www.rfc-editor.org/info/rfc3183>.

[RFC3207] Hoffman, P., "SMTP Service Extension for Secure SMTP over Transport Layer Security", RFC 3207, February 2002, <http://www.rfc-editor.org/info/rfc3207>.

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003, <http://www.rfc-editor.org/info/rfc3501>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, October 2008, <http://www.rfc-editor.org/info/rfc5321>.

[RFC5598] Crocker, D., "Internet Mail Architecture", RFC 5598, July 2009, <http://www.rfc-editor.org/info/rfc5598>.

[RFC5750] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Certificate Handling", RFC 5750, January 2010, <http://www.rfc-editor.org/info/rfc5750>.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010, <http://www.rfc-editor.org/info/rfc5751>.

Appendix A. Formal Syntax of Secure Header

Note: The ASN.1 module contained herein uses the 1988 version of ASN.1 notation [ASN1-88] for the purposes of alignment with the existing S/MIME specifications. The SecureHeaderFields structure is defined as follows:

```
mod-SMimeSecureHeadersV1
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
  pkcs-9(9) smime(16) modules(0) secure-headers-v1(65) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

IMPORTS

  id-aa
       FROM SecureMimeMessageV3dot1
            { iso(1) member-body(2) us(840) rsadsi(113549)
            pkcs(1) pkcs-9(9) smime(16) modules(0)
            msg-v3dot1(21) };

-- id-aa is the arc with all new authenticated and
-- unauthenticated attributes produced by the S/MIME
-- Working Group

 id-aa-secureHeaderFieldsIdentifier OBJECT IDENTIFIER ::= {
    id-aa secure-headers(55) }

 SecureHeaderFields ::= SET {
      canonAlgorithm Algorithm,
      secHeaderFields HeaderFields }

 Algorithm ::= ENUMERATED {
      canonAlgorithmSimple(0),
      canonAlgorithmRelaxed(1) }

 HeaderFields ::= SEQUENCE SIZE (1..MAX) OF HeaderField

 HeaderField ::= SEQUENCE {
      field-Name HeaderFieldName,
      field-Value HeaderFieldValue,
      field-Status HeaderFieldStatus DEFAULT duplicated }

 HeaderFieldName ::= VisibleString (FROM (ALL EXCEPT (":")))
      -- This description matches with the description of
      -- field name in the Sections 2.2 and 3.6.8 of RFC 5322
```

```
HeaderFieldValue ::= UTF8String
     -- This description matches with the description of
     -- field body in the Section 2.2 of RFC 5322 as
     -- extended by Section 3.1 of RFC 6532.

HeaderFieldStatus ::= INTEGER {
     duplicated(0), deleted(1), modified(2) }

END
```

Appendix B. Example of Secure Header Fields

In the following example, the header fields subject, x-ximf-primary-precedence, and x-ximf-correspondance-type are secured and integrated in a SecureHeaderFields structure. This example should produce a valid signature.

Extract from the message header fields:

```
From: John Doe <jdoe@example.com>
To: Mary Smith <mary@example.com>
subject: This is a test of Ext.
x-ximf-primary-precedence: priority
x-ximf-correspondance-type: official
```

The SecureHeaderFields structure extracted from the signature:

```
2286  150: SEQUENCE {
2289   11:   OBJECT IDENTIFIER '1 2 840 113549 1 9 16 2 80'
2302  134:   SET {
2305  131:     SET {
2308    4:       ENUMERATED 1
2314  123:       SEQUENCE {
2316   40:         SEQUENCE {
2318   25:           VisibleString 'x-ximf-primary-precedence'
2345    8:           UTF8String 'priority'
2355    1:           INTEGER 0
         :           }
2358   41:         SEQUENCE {
2360   26:           VisibleString 'x-ximf-correspondance-type'
2388    8:           UTF8String 'official'
2398    1:           INTEGER 0
         :           }
2401   36:         SEQUENCE {
2403    7:           VisibleString 'subject'
2412   22:           UTF8String 'This is a test of Ext.'
2436    1:           INTEGER 0
         :           }
         :         }
         :       }
         :     }
         :   }
```

The example is displayed as an output of Peter Gutmann's "dumpasn1" program.

OID used in this example is nonofficial.

Acknowledgements

The authors would like to thank Jim Schaad, Alexey Melnikov, Damien Roque, Thibault Cassan, William Ottaway, and Sean Turner who kindly provided reviews of the document and/or suggestions for improvement. As always, all errors and omissions are the responsibility of the authors.

Authors' Addresses

Laurent CAILLEUX
DGA MI
BP 7
35998 RENNES CEDEX 9
France

EMail: laurent.cailleux@intradef.gouv.fr

Chris Bonatti
IECA, Inc.
3057 Nutley Street, Suite 106
Fairfax, VA 22031
United States

EMail: bonatti252@ieca.com