Network Working Group
Request for Comments: 3156
Updates: 2015
Category: Standards Track

M. Elkins
Network Associates, Inc.
D. Del Torto
CryptoRights Foundation
R. Levien
University of California at Berkeley
T. Roessler

August 2001

MIME Security with OpenPGP

Status of this Memo

Copyright Notice

Abstract

This document describes how the OpenPGP Message Format can be used to provide privacy and authentication using the Multipurpose Internet Mail Extensions (MIME) security content types described in RFC 1847.

1. Introduction

Work on integrating PGP (Pretty Good Privacy) with MIME [3] (including the since withdrawn "application/pgp" content type) prior to RFC 2015 suffered from a number of problems, the most significant of which is the inability to recover signed message bodies without parsing data structures specific to PGP. RFC 2015 makes use of the elegant solution proposed in RFC 1847, which defines security multipart formats for MIME. The security multiparts clearly separate the signed message body from the signature, and have a number of other desirable properties. This document revises RFC 2015 to adopt the integration of PGP and MIME to the needs which emerged during the work on the OpenPGP specification.

This document defines three content types for implementing security and privacy with OpenPGP: "application/pgp-encrypted", "application/pgp-signature" and "application/pgp-keys".

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 RFC 2119.

2. OpenPGP data formats

OpenPGP implementations can generate either ASCII armor (described in [1]) or 8-bit binary output when encrypting data, generating a digital signature, or extracting public key data. The ASCII armor output is the REQUIRED method for data transfer. This allows those users who do not have the means to interpret the formats described in this document to be able to extract and use the OpenPGP information in the message.

When the amount of data to be transmitted requires that it be sent in many parts, the MIME message/partial mechanism SHOULD be used rather than the multi-part ASCII armor OpenPGP format.

3. Content-Transfer-Encoding restrictions

Multipart/signed and multipart/encrypted are to be treated by agents as opaque, meaning that the data is not to be altered in any way [2], [7]. However, many existing mail gateways will detect if the next hop does not support MIME or 8-bit data and perform conversion to either Quoted-Printable or Base64. This presents serious problems for multipart/signed, in particular, where the signature is invalidated when such an operation occurs. For this reason all data signed according to this protocol MUST be constrained to 7 bits (8-bit data MUST be encoded using either Quoted-Printable or Base64). Note that this also includes the case where a signed object is also encrypted (see section 6). This restriction will increase the likelihood that the signature will be valid upon receipt.

Additionally, implementations MUST make sure that no trailing whitespace is present after the MIME encoding has been applied.

> Note: In most cases, trailing whitespace can either be removed, or protected by applying an appropriate content-transfer-encoding. However, special care must be taken when any header lines - either in MIME entity headers, or in embedded RFC 822 headers - are present which only consist of whitespace: Such lines must be removed entirely, since replacing them by empty lines would turn them into header delimiters, and change the semantics of the message. The restrictions on whitespace are necessary in order to make the hash calculated invariant under the text and binary mode signature mechanisms provided by OpenPGP [1]. Also, they help to avoid compatibility problems with PGP implementations which predate the OpenPGP specification.

Note: If any line begins with the string "From ", it is strongly suggested that either the Quoted-Printable or Base64 MIME encoding be applied. If Quoted-Printable is used, at least one of the characters in the string should be encoded using the hexadecimal coding rule. This is because many mail transfer and delivery agents treat "From " (the word "from" followed immediately by a space character) as the start of a new message and thus insert a right angle-bracket (>) in front of any line beginning with "From " to distinguish this case, invalidating the signature.

Data that is ONLY to be encrypted is allowed to contain 8-bit characters and trailing whitespace and therefore need not undergo the conversion to a 7bit format, and the stripping of whitespace.

Implementor's note: It cannot be stressed enough that applications using this standard follow MIME's suggestion that you "be conservative in what you generate, and liberal in what you accept." In this particular case it means it would be wise for an implementation to accept messages with any content-transfer-encoding, but restrict generation to the 7-bit format required by this memo. This will allow future compatibility in the event the Internet SMTP framework becomes 8-bit friendly.

4. OpenPGP encrypted data

Before OpenPGP encryption, the data is written in MIME canonical format (body and headers).

OpenPGP encrypted data is denoted by the "multipart/encrypted" content type, described in [2], and MUST have a "protocol" parameter value of "application/pgp-encrypted". Note that the value of the parameter MUST be enclosed in quotes.

The multipart/encrypted MIME body MUST consist of exactly two body parts, the first with content type "application/pgp-encrypted". This body contains the control information. A message complying with this standard MUST contain a "Version: 1" field in this body. Since the OpenPGP packet format contains all other information necessary for decrypting, no other information is required here.

The second MIME body part MUST contain the actual encrypted data. It MUST be labeled with a content type of "application/octet-stream".

Example message:

```
From: Michael Elkins <elkins@aero.org>
To: Michael Elkins <elkins@aero.org>
Mime-Version: 1.0
```

```
Content-Type: multipart/encrypted; boundary=foo;
   protocol="application/pgp-encrypted"

--foo
Content-Type: application/pgp-encrypted

Version: 1

--foo
Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----
Version: 2.6.2

hIwDY32hYGCE8MkBA/wOu7d45aUxF4Q0RKJprD3v5Z9K1YcRJ2fve87lMlDlx4Oj
eW4GDdBfLbJE7VUpp13N19GL8e/AqbyyjHH4aS0YoTk10QQ9nnRvjY8nZL3MPXSZ
g9VGQxFeGqzykzmykU6A26MSMexR4ApeeON6xzZWfo+0yOqAq6lb46wsvldZ96YA
AABH78hyX7YX4uT1tNCWEIIBoqqvCeIMpp7UQ2IzBrXg6GtukS8NxbukLeamqVW3
1yt21DYOjuLzcMNe/JNsD9vDVCvOOG3OCi8=
=zzaA
-----END PGP MESSAGE-----

--foo--
```

5. OpenPGP signed data

OpenPGP signed messages are denoted by the "multipart/signed" content type, described in [2], with a "protocol" parameter which MUST have a value of "application/pgp-signature" (MUST be quoted).

The "micalg" parameter for the "application/pgp-signature" protocol MUST contain exactly one hash-symbol of the format "pgp-<hash-identifier>", where <hash-identifier> identifies the Message Integrity Check (MIC) algorithm used to generate the signature. Hash-symbols are constructed from the text names registered in [1] or according to the mechanism defined in that document by converting the text name to lower case and prefixing it with the four characters "pgp-".

Currently defined values are "pgp-md5", "pgp-sha1", "pgp-ripemd160", "pgp-md2", "pgp-tiger192", and "pgp-haval-5-160".

The multipart/signed body MUST consist of exactly two parts. The first part contains the signed data in MIME canonical format, including a set of appropriate content headers describing the data.

The second body MUST contain the OpenPGP digital signature. It MUST be labeled with a content type of "application/pgp-signature".

Note: Implementations can either generate "signatures of a canonical text document" or "signatures of a binary document", as defined in [1]. The restrictions on the signed material put forth in section 3 and in this section will make sure that the various MIC algorithm variants specified in [1] and [5] will all produce the same result.

When the OpenPGP digital signature is generated:

(1) The data to be signed MUST first be converted to its content-type specific canonical form. For text/plain, this means conversion to an appropriate character set and conversion of line endings to the canonical <CR><LF> sequence.

(2) An appropriate Content-Transfer-Encoding is then applied; see section 3. In particular, line endings in the encoded data MUST use the canonical <CR><LF> sequence where appropriate (note that the canonical line ending may or may not be present on the last line of encoded data and MUST NOT be included in the signature if absent).

(3) MIME content headers are then added to the body, each ending with the canonical <CR><LF> sequence.

(4) As described in section 3 of this document, any trailing whitespace MUST then be removed from the signed material.

(5) As described in [2], the digital signature MUST be calculated over both the data to be signed and its set of content headers.

(6) The signature MUST be generated detached from the signed data so that the process does not alter the signed data in any way.

Note: The accepted OpenPGP convention is for signed data to end with a <CR><LF> sequence. Note that the <CR><LF> sequence immediately preceding a MIME boundary delimiter line is considered to be part of the delimiter in [3], 5.1. Thus, it is not part of the signed data preceding the delimiter line. An implementation which elects to adhere to the OpenPGP convention has to make sure it inserts a <CR><LF> pair on the last line of the data to be signed and transmitted (signed message and transmitted message MUST be identical).

Example message:

```
From: Michael Elkins <elkins@aero.org>
To: Michael Elkins <elkins@aero.org>
Mime-Version: 1.0
```

```
   Content-Type: multipart/signed; boundary=bar; micalg=pgp-md5;
     protocol="application/pgp-signature"

   --bar
& Content-Type: text/plain; charset=iso-8859-1
& Content-Transfer-Encoding: quoted-printable
&
& =A1Hola!
&
& Did you know that talking to yourself is a sign of senility?
&
& It’s generally a good idea to encode lines that begin with
& From=20because some mail transport agents will insert a greater-
& than (>) sign, thus invalidating the signature.
&
& Also, in some cases it might be desirable to encode any   =20
& trailing whitespace that occurs on lines in order to ensure  =20
& that the message signature is not invalidated when passing =20
& a gateway that modifies such whitespace (like BITNET). =20
&
& me

--bar

Content-Type: application/pgp-signature

-----BEGIN PGP MESSAGE-----
Version: 2.6.2

iQCVAwUBMJrRF2N9oWBghPDJAQE9UQQAtl7LuRVndBjrk4EqYBIb3h5QXIX/LC//
jJV5bNvkZIGPIcEmI5iFd9boEgvpirHtIREEqLQRkYNoBActFBZmh9GC3C041WGq
uMbrbxc+nIs1TIKlA08rVi9ig/2Yh7LFrK5Ein57U/W72vgSxLhe/zhdfolT9Brn
HOxEa44b+EI=
=ndaj
-----END PGP MESSAGE-----

--bar--
```

The "&"s in the previous example indicate the portion of the data over which the signature was calculated.

Upon receipt of a signed message, an application MUST:

(1) Convert line endings to the canonical <CR><LF> sequence before the signature can be verified. This is necessary since the local MTA may have converted to a local end of line convention.

(2) Pass both the signed data and its associated content headers along with the OpenPGP signature to the signature verification service.

6. Encrypted and Signed Data

Sometimes it is desirable to both digitally sign and then encrypt a message to be sent. This protocol allows for two methods of accomplishing this task.

6.1. RFC 1847 Encapsulation

In [2], it is stated that the data is first signed as a multipart/signature body, and then encrypted to form the final multipart/encrypted body. This is most useful for standard MIME-compliant message forwarding.

Example:

```
      Content-Type: multipart/encrypted;
         protocol="application/pgp-encrypted"; boundary=foo

      --foo
      Content-Type: application/pgp-encrypted

      Version: 1

      --foo
      Content-Type: application/octet-stream

      -----BEGIN PGP MESSAGE-----
   & Content-Type: multipart/signed; micalg=pgp-md5
   &     protocol="application/pgp-signature"; boundary=bar
   &
   & --bar
   & Content-Type: text/plain; charset=us-ascii
   &
   & This message was first signed, and then encrypted.
   &
   & --bar
   & Content-Type: application/pgp-signature
   &
   & -----BEGIN PGP MESSAGE-----
   & Version: 2.6.2
   &
   & iQCVAwUBMJrRF2N9oWBghPDJAQE9UQQAtl7LuRVndBjrk4EqYBIb3h5QXIX/LC//
   & jJV5bNvkZIGPIcEmI5iFd9boEgvpirHtIREEqLQRkYNoBActFBZmh9GC3C041WGq
   & uMbrbxc+nIs1TIKlA08rVi9ig/2Yh7LFrK5Ein57U/W72vgSxLhe/zhdfolT9Brn
```

```
& HOxEa44b+EI=
& =ndaj
& -----END PGP MESSAGE-----
&
& --bar--
  -----END PGP MESSAGE-----

  --foo--
```

(The text preceded by '&' indicates that it is really encrypted, but presented as text for clarity.)

6.2. Combined method

The OpenPGP packet format [1] describes a method for signing and encrypting data in a single OpenPGP message. This method is allowed in order to reduce processing overhead and increase compatibility with non-MIME implementations of OpenPGP. The resulting data is formatted as a "multipart/encrypted" object as described in Section 4.

Messages which are encrypted and signed in this combined fashion are REQUIRED to follow the same canonicalization rules as multipart/signed objects.

It is explicitly allowed for an agent to decrypt a combined message and rewrite it as a multipart/signed object using the signature data embedded in the encrypted version.

7. Distribution of OpenPGP public keys

Content-Type: application/pgp-keys
Required parameters: none
Optional parameters: none

A MIME body part of the content type "application/pgp-keys" contains ASCII-armored transferable Public Key Packets as defined in [1], section 10.1.

8. Security Considerations

Signatures of a canonical text document as defined in [1] ignore trailing white space in signed material. Implementations which choose to use signatures of canonical text documents will not be able to detect the addition of whitespace in transit.

See [3], [4] for more information on the security considerations concerning the underlying protocols.

9. IANA Considerations

This document defines three media types: "application/pgp-encrypted", "application/pgp-signature" and "application/pgp-keys". The following sections specify the IANA registrations for these types.

9.1. Registration of the application/pgp-encrypted media type

MIME media type name: application
MIME subtype name: pgp-encrypted
Required parameters: none
Optional parameters: none

Encoding considerations:

Currently this media type always consists of a single 7bit text string.

Security considerations:

See Section 8 and RFC 2440 Section 13.

Interoperability considerations: none

Published specification:

This document.

Additional information:

Magic number(s): none
File extension(s): none
Macintosh File Type Code(s): none

Person & email address to contact for further information:

Michael Elkins
Email: me@cs.hmc.edu

Intended usage: common

Author/Change controller:

Michael Elkins
Email: me@cs.hmc.edu

9.2. Registration of the application/pgp-signature media type

MIME media type name: application
MIME subtype name: pgp-signature
Required parameters: none
Optional parameters: none

Encoding considerations:

The content of this media type always consists of 7bit text.

Security considerations:

See Section 8 and RFC 2440 Section 13.

Interoperability considerations: none

Published specification:

RFC 2440 and this document.

Additional information:

Magic number(s): none
File extension(s): asc, sig
Macintosh File Type Code(s): pgDS

Person & email address to contact for further information:

Michael Elkins
Email: me@cs.hmc.edu

Intended usage: common

Author/Change controller:

Michael Elkins
Email: me@cs.hmc.edu

9.3. Registration of the application/pgp-keys media type

MIME media type name: application
MIME subtype name: pgp-keys
Required parameters: none
Optional parameters: none

Encoding considerations:

The content of this media type always consists of 7bit text.

Security considerations:

See Section 8 and RFC 2440 Section 13.

Interoperability considerations: none

Published specification:

RFC 2440 and this document.

Additional information:

Magic number(s): none
File extension(s): asc
Macintosh File Type Code(s): none

Person & email address to contact for further information:

Michael Elkins
Email: me@cs.hmc.edu

Intended usage: common

Author/Change controller:

Michael Elkins
Email: me@cs.hmc.edu

10. Notes

"PGP" and "Pretty Good Privacy" are registered trademarks of Network Associates, Inc.

11. Acknowledgements

This document relies on the work of the IETF's OpenPGP Working Group's definitions of the OpenPGP Message Format. The OpenPGP message format is currently described in RFC 2440 [1].

Special thanks are due: to Philip Zimmermann for his original and ongoing work on PGP; to Charles Breed, Jon Callas and Dave Del Torto for originally proposing the formation of the OpenPGP Working Group; and to Steve Schoenfeld for helpful feedback during the draft process. The authors would also like to thank the engineers at Pretty Good Privacy, Inc (now Network Associates, Inc), including Colin Plumb, Hal Finney, Jon Callas, Mark Elrod, Mark Weaver and Lloyd Chambers, for their technical commentary.

Additional thanks are due to Jeff Schiller and Derek Atkins for their continuing support of strong cryptography and PGP freeware at MIT; to Rodney Thayer of Sable Technology; to John Noerenberg, Steve Dorner and Laurence Lundblade of the Eudora team at QUALCOMM, Inc; to Bodo Moeller for proposing the approach followed with respect to trailing whitespace; to John Gilmore, Hugh Daniel and Fred Ringel (at Rivertown) and Ian Bell (at Turnpike) for their timely critical commentary; and to the international members of the IETF's OpenPGP mailing list, including William Geiger, Lutz Donnerhacke and Kazu Yamamoto. The idea to use multipart/mixed with multipart/signed has been attributed to James Galvin. Finally, our gratitude is due to the many members of the "Cypherpunks," "Coderpunks" and "pgp-users" <http://cryptorights.org/pgp-users> mailing lists and the many users of PGP worldwide for helping keep the path to privacy open.

12. Addresses of the Authors and OpenPGP Working Group Chair

The OpenPGP working group can be contacted via the current chair:

John W. Noerenberg II
Qualcomm, Inc.
5775 Morehouse Dr.
San Diego, CA 92121 USA

Phone: +1 619 658 3510
EMail: jwn2@qualcomm.com

The principal authors of this document are:

Dave Del Torto
CryptoRights Foundation
80 Alviso Street, Mailstop: CRF
San Francisco, CA 94127 USA

Phone: +1.415.334.5533, vm: #2
EMail: ddt@cryptorights.org, ddt@openpgp.net

Michael Elkins
Network Associates, Inc.
3415 S. Sepulveda Blvd Suite 700
Los Angeles, CA 90034 USA

Phone: +1.310.737.1663
Fax: +1.310.737.1755
Email: me@cs.hmc.edu, Michael_Elkins@NAI.com

Raph Levien
University of California at Berkeley
579 Soda Hall
Berkeley, CA 94720 USA

Phone: +1.510.642.6509
EMail: raph@acm.org

Thomas Roessler
Nordstrasse 99
D-53111 Bonn, Germany

Phone: +49-228-638007
EMail: roessler@does-not-exist.org

References

[1] Callas, J., Donnerhacke, L., Finney, H. and R. Thayer, "OpenPGP Message Format", RFC 2440, November 1998.

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

[3] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[4] Galvin, J., Murphy, G., Crocker, S. and N. Freed, "MIME Object Security Services", RFC 1848, October 1995.

[5] Atkins, D., Stallings, W. and P. Zimmermann, "PGP Message Exchange Formats", RFC 1991, August 1996.

[6] Elkins, M., "MIME Security with Pretty Good Privacy (PGP)", RFC 2015, October 1996.

[7] Freed, N., "Gateways and MIME Security Multiparts", RFC 2480, January 1999.

Full Copyright Statement

Acknowledgement

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