Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 13, 2019

C. Luck
pEp Foundation
B. Hoeneisen
Ucom.ch
March 12, 2019

pretty Easy privacy (pEp): Header Protection
draft-luck-lamps-pep-header-protection-01

Abstract

Issues with email header protection in S/MIME have been recently raised in the IETF LAMPS Working Group. The need for amendments to the existing specification regarding header protection was expressed.

The pretty Easy privacy (pEp) implementations currently use a mechanism quite similar to the currently standardized message wrapping for S/MIME. The main difference is that pEp is using PGP/MIME instead, and adds space for carrying public keys next to the protected message.

In LAMPS voices have also been expressed, that whatever mechanism will be chosen, it should not be limited to S/MIME, but also applicable to PGP/MIME.

This document aims to contribute to this discussion and share pEp implementation experience with email header protection.

Table of Contents

1. Introduction

A range of protocols for the protection of electronic mail (email) exist, which allow to assess the authenticity and integrity of the email headers section or selected header fields from the domain-level perspective, specifically DomainKeys Identified Mail (DKIM) [RFC6376] and Sender Policy Framework (SPF) [RFC7208] and Domain-based Message Authentication, Reporting, and Conformance (DMARC) [RFC7489]. These protocols, while essential to responding to a range of attacks on email, do not offer full end-to-end protection to the headers section and are not capable of providing privacy for the information contained therein.

The need for means of Data Minimization, which includes data spareness and hiding of all information, which technically can be hidden, has grown in importance over the past years.

A standard for end-to-end protection of the email headers section exists for S/MIME since version 3.1. (cf. [RFC5751] and [I-D.ietf-lamps-rfc5751-bis]):

> In order to protect outer, non-content-related message header fields (for instance, the "Subject", "To", "From", and "Cc" fields), 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.

No mechanism for header protection has been standardized for PGP (Pretty Good Privacy) yet.

End-to-end protection for the email headers section is currently not widely implemented - neither for messages protected by means of S/MIME nor PGP. At least two variants of header protection are known to be implemented. A recently submitted Internet-Draft [I-D.melnikov-lamps-header-protection] discusses the two variants and the challenges with header protection for S/MIME. The two variants are referred to as:

- Option 1: Memory Hole
- Option 2: Wrapping with message/rfc822 or message/global

pEp (pretty Easy privacy) [I-D.birk-pep] for email [I-D.marques-pep-email] already implements an option quite similar to Option 2, adapting the S/MIME standards to PGP/MIME (cf. Section 5, ff.). Existing implementations of pEp have also added inbound support for "Memory Hole" referred to above as Option 1, thus being able to study the differences and the implementator's challenges.

Interoperability and implementation symmetry between PGP/MIME and S/MIME is planned by pEp, but still in an early stage of development.

This document lists generic use cases (Section 3) and requirements for header protection (Section 4) and describes progressive header disclosure as implemented in the "pEp message format version 2". This format inherently offers header protection, and may be implemented independently by mail user agents otherwise not conforming to pEp standards (Section 5, ff.).

2. Terms

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].

- Man-in-the-middle attack (MITM): cf. [RFC4949]

2.1. The OpenPGP Radix-64

In the examples following in this section, it is a common pattern to have a MIME encoded mail containing ("wrapping") another signed and eventually encrypted mail. Such enclosed mails are encoded following the OpenPGP standard, which specifies an encoding called "Radix-64", which is 7-bit transport-encoding compatible by design.

The Radix-64 consists of a begin and an end Armor Header Line, a stream of base64-encoded data limited to 78 characters per line plus <CR><LF>, and an encoded CRC-24 value.

The following is an example, with some similar lines of base64 output replaced with ellipsis:

```
-----BEGIN PGP MESSAGE-----
hQIMAwusnBHN80H+AQ//cJLQLOl+6hOofKEkQJeu0wedmwt+TkzPx/sCUQ80dzLv
...
j/ES8ndDBftM5mZLzFQ2VatqB9G9cqCgiOVFs6jfTI13nPfLit9IPWRavcVIMdwt
Xd9bdvHx/ReenAk/
=7WaL
-----END PGP MESSAGE-----
```

To make the examples look more compact and relevant, the above will be replaced symbolically by:

```
[[----- OpenPGP Radix-64 Block -----]]
```

2.1.1. Radix-64 in the Context of MIME Messages

Note that OpenPGP and MIME specifications overlap when a Radix-64 immediately precedes a MIME boundary. The <CR><LF> sequence immediately preceding a MIME boundary delimiter line is considered to be part of the delimiter in [RFC2046], 5.1. And in OpenPGP, line endings are considered a part of the Armor Header Line for the purposes of determining the content they delimit in [RFC4880], 6.2. Keeping an empty line between the end Armor Header Line and the MIME boundary line is suggested.

3. Use Cases

In the following, we show the generic use cases that need to be addressed independently of whether S/MIME, PGP/MIME or any other technology is used for which Header Protection (HP) is to be applied to.

3.1. Interactions

The main interaction case for Header Protection (HP) is:

1) Both peers (sending and receiving side) fully support HP

For backward compatibility of legacy clients - unaware of any HP - the following intermediate interactions need to be considered as well:

2) The sending side fully supports HP, while the receiving side does not support any HP

3) The sending side does not support any HP, while the receiving side fully supports HP (trivial case)

4) Neither the sending side nor the receiving side supports any HP (trivial case)

The following intermediate use cases may need to be considered as well for backward compatibility with legacy HP systems, such as S/MIME since version 3.1 (cf. [RFC5751] and [I-D.ietf-lamps-rfc5751-bis]), in the following designated as legacy HP:

5) The sending side fully supports HP, while the receiving side supports legacy HP only

6) The sending side supports legacy HP only, while the receiving side fully supports HP

7) Both peers (sending and receiving side) support legacy HP only

8) The sending side supports legacy HP only, while the receiving side does not support any HP

9) The sending side does not support any HP, while the receiving side supports legacy HP only (trivial case)

Note: It is to be decided whether to ensure legacy HP systems do not conflict with any new solution for HP at all or whether (and to which degree) backward compatibility to legacy HP systems shall be maintained.

3.2. Protection Levels

The following protection levels need to be considered:

a) signature and encryption

b) signature only

c) encryption only [[TODO: verify whether relevant]]

4. Requirements

In the following a list of requirements that need to be addressed independently of whether S/MIME, PGP/MIME or any other technology is used to apply HP to.

4.1. General Requirements

This subsection is listing the requirements to address use case 1) (cf. Section 3.1).

G1: Define the format for HP for all protection levels. This includes MIME structure, Content-Type (including charset and name), Content-Disposition (including filename), and Content-Transfer-Encoding. Furthermore, it must be defined, how a public key should be included.

G3: To foster wide implementation of the new solution, it shall be easily implementable. Unless needed for maximizing protection and privacy, existing implementations shall not require substantial changes in the existing code base. In particular also MIME libraries widely used shall not need to be changed to comply with the new mechanism for HP.

G4: Ensure that man-in-the-middle attack (MITM) cf. {{RFC4949}}, in particular downgrade attacks, are mitigated as good as possible.

4.1.1. Sending Side

GS1: Determine which Header Fields (HFs) should or must be protected at least for all protection levels.

GS2: Determine which HFs should or must be sent in clear of an encrypted email.

GS3: Determine which HF should not or must not be included in the visible header (for transport) of an encrypted email, with the default being that whatever is not needed from GS2 is not put into the unencrypted transport headers, thus fulfilling data minimization requirements (including data spareness and hiding of all information that technically can be hidden).

4.1.2. Receiving Side

GR1: Determine how HF should be displayed to the user in case of conflicting information between the protected and unprotected headers.

GR2: Ensure that man-in-the-middle attack (MITM) cf. {{RFC4949}}, in particular downgrade attacks, can be detected.

4.2. Additional Requirements for Backward-Compatibility With Legacy Clients Unaware of Header Protection

This sub-section addresses the use cases 2) - 4) (cf. Section 3.1)

B1: Depending on the solution, define a means to distinguish between forwarded messages and encapsulated messages using new HP mechanism.

4.2.1. Sending side

BS1: Define how full HP support can be indicated to outgoing messages.

BS2: Define how full HP support of the receiver can be detected or guessed.

BS3: Ensure a HP unaware receiving side easily can display the "Subject" HF to the user.

4.2.2. Receiving side

BR1: Define how full HP support can be detected in incoming messages.

4.3. Additional Requirements for Backward-Compatibility with Legacy Header Protection Systems (if supported)

This sub-section addresses the use cases 5) - 9) (cf. Section 3.1).

LS1: Depending on the solution, define a means to distinguish between forwarded messages, legacy encapsulated messages, and encapsulated messages using new HP mechanism.

LS2: The solution should be backward compatible to existing solutions and aim to minimize the implementation effort to include support for existing solutions.

4.3.1. Sending Side

LSS1: Determine how legacy HP support can be indicated to outgoing messages.

LSS2: Determine how legacy HP support of the receiver can be detected or guessed.

4.3.2. Receiving Side

LSR1: Determine how legacy HP support can be detected in incoming messages.

5. Message Format for progressive header disclosure

5.1. Design principles

pretty Easy privacy (pEp) is working on bringing state-of-the-art automatic cryptography known from areas like TLS to electronic mail (email) communication. pEp is determined to evolve the existing standards as fundamentally and comprehensively as needed to gain easy implementation and integration, and for easy use for regular Internet users. pEp for email wants to attaining to good security practice while still retaining backward compatibility for implementations widespread.

To provide the required stability as a foundation for good security practice, pEp for email defines a fixed MIME structure for its innermost message structure, so to remove most attack vectors which have permitted the numerous EFAIL vulnerabilities. (TBD: ref)

Security comes just next after privacy in pEp, for which reason the application of signatures without encryption to messages in transit is not considered purposeful. pEp for email herein referenced, and further described in [I-D.marques-pep-email], either expects to transfer messages in cleartext without signature or encryption, or transfer them encrypted and with enclosed signature and necessary

public keys so that replies can be immediately upgraded to encrypted messages.

The pEp message format is equivalent to the S/MIME standard in ensuring header protection, in that the whole message is protected instead, by wrapping it and providing cryptographic services to the whole original message. The pEp message format is different compared to the S/MIME standard in that the pEp protocols propose opportunistic end-to-end security and signature, by allowing the transport of the necessary public key material along with the original messages.

For the purpose of allowing the insertion of such public keys, the root entity of the protected message is thus nested once more into an additional multipart/mixed MIME entity. The current pEp proposal is for PGP/MIME, while an extension to S/MIME is next.

pEp's proposal is strict in that it requires that the cryptographic services applied to the protected message MUST include encryption. It also mandates a fixed MIME structure for the protected message, which always MUST include a plaintext and optionally an HTML representation (if HTML is used) of the same message, and requires that all other optional elements to be eventually presented as attachments. Alternatively the whole protected message could represent in turn a wrapped pEp wrapper, which makes the message structure fully recursive on purpose (e.g., for the purpose of anonymization through onion routing).

For the purpose of implementing mixnet routing for email, it is foreseen to nest pEp messages recursively. A protected message can in turn contain a protected message due for forwarding. This is for the purpose to increase privacy and counter the necessary leakage of plaintext addressing in the envelope of the email.

The recursive nature of the pEp message format allows for the implementation of progressive disclosure of the necessary transport relevant header fields just as-needed to the next mail transport agents along the transmission path.

5.2. Compatibility

The pEp message format version 2 is designed such that a receiving Mail User Agent (MUA), which is OpenPGP-compliant but not pEp-compliant, still has built-in capability to properly verify the integrity of the mail, decode it and display all information of the original mail to the user. The recovered protected message is selfsufficiently described, including all protected header fields.

The pEp message format version 2 (as used by all the various pEp implementations, cf. Section 10) is similar to what is standardized for S/MIME in [RFC5751] and its successor [I-D.ietf-lamps-rfc5751-bis]:

> In order to protect outer, non-content-related message header fields (for instance, the "Subject", "To", "From", and "Cc" fields), 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.
>
> When an S/MIME message is received, if the top-level protected MIME entity has a Content-Type of message/rfc822, it can be assumed that the intent was to provide header protection. This entity SHOULD be presented as the top-level message, [...].

5.3. Inner message

The pEp message format requires the innermost protected message to follow a fixed MIME structure and to consist of exactly one human-readable message which is represented in plain or HTML format. Both plain and html entities MUST represent the same message to the user. Any attachment to the message must be laid out in a flat list. No additional multipart entities are allowed in the pEp message.

These restrictions permit to build mail user agents which are immune to the EFAIL attacks.

This message is herein further referred to as the "pEp inner message".

A mail user agent wanting to follow this standard, SHOULD transform any "original message" into a "pEp inner message" for safe representation on the receiving side.

5.4. Content-Type property "forwarded"

One caveat of the design is that the user interaction with message/rfc822 entities varies considerably across different mail user agents. No standard is currently available which enables MUAs to reliably determine whenever a nested message/rfc822 entity is meant to blend the containing message, or if it was effectively intended to be forwarded as a file document. pEp currently intends to implement the proposal described by [I-D.melnikov-lamps-header-protection], 3.2, which defines a new Content-Type header field parameter with name "forwarded", for the MUA to distinguish between a forwarded

message and a nested message for the purpose of header protection, i.e., using "forwarded=no".

5.5. Outer message

With pEp message format version 2, the pEp standardized message is equally wrapped in a message/rfc822 entity, but this time being in turn wrapped in a multipart/mixed entity. The purpose of the additional nesting is to allow for public keys of the sender to be stored alongside the original message while being protected by the same mechanism.

For the case of PGP/MIME, the currently only implemented MIME encryption protocol implemented in pEp, the top-level entity called the "outer message" MUST contain:

- exactly one entity of type message/rfc822, and
- one or more entity of type application/pgp-keys

Notes on the current pEp client implementations:

- the current pEp implementation also adds a text/plain entity containing "pEp-Wrapped-Message-Info: OUTER" as first element in the MIME tree. This element is not strictly necessary, but is in place for better backwards compatibility when manually navigating the nested message structure. This is part of the study of various solutions to maximize backwards compatibility, and has been omitted from the following examples.
- the current pEp implementation prepends "pEp-Wrapped-Message-Info: INNER<CR><LF>" to the original message body. This is an implementation detail which should be ignored, and has been omitted in the following examples.
- the current pEp implementation may render a text/plain directly in place of the multipart/alternate, when no HTML representation was generated by the sending MUA. This is not strict according to pEp's own specification, and is currently being investigated.

This is an example of the top-level MIME entity, before being encrypted and signed:

```
MIME-Version: 1.0
Content-Type: multipart/mixed;
              boundary="6b8b4567327b23c6643c986966334873"


--6b8b4567327b23c6643c986966334873
Content-Type: message/rfc822; forwarded="no"

From: John Doe <jdoe@machine.example>
To: Mary Smith <mary@example.net>
Subject: Example
Date: Fri, 30 Jun 2018 09:55:06 +0200
Message-ID: <05d0526e-41c4-11e9-8828@pretty.Easy.privacy>
X-Pep-Version: 2.0
MIME-Version: 1.0
Content-Type: multipart/alternative;
              boundary="29fe9d2b2d7f6a703c1bffc47c162a8c"

--29fe9d2b2d7f6a703c1bffc47c162a8c
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: quoted-printable
Content-Disposition: inline; filename="msg.txt"

p=E2=89=A1p for Privacy by Default.
-- =20
Sent from my p=E2=89=A1p for Android.

--29fe9d2b2d7f6a703c1bffc47c162a8c
Content-Type: text/html; charset="utf-8"
Content-Transfer-Encoding: quoted-printable

p=E2=89=A1p for Privacy by Default.<br>
-- <br>
Sent from my p=E2=89=A1p for Android.<br>

--29fe9d2b2d7f6a703c1bffc47c162a8c--
--6b8b4567327b23c6643c986966334873
Content-Type: application/pgp-keys
Content-Disposition: attachment; filename="pEpkey.asc"

-----BEGIN PGP PUBLIC KEY BLOCK-----

...
-----END PGP PUBLIC KEY BLOCK-----

--6b8b4567327b23c6643c986966334873--
```

5.6. Transport message

In pEp message format 2 the "outer message" consists of a full RFC822 message with body and a minimal set of header fields, just those necessary to conform to MIME multipart standards.

The "outer message" should be encrypted and carry a signature according to the MIME encryption standards. The resulting message is the transport message which a root entity of type multipart/encrypted.

A minimal set of header fields should be set on the "transport message", as to permit delivery, without disclosing private information.

The structure of the transport message may be altered in-transit, e.g. through mailing list agents, or inspection gateways.

Signing and encrypting a message with MIME Security with OpenPGP [RFC3156], yields a message with the following complete MIME structure, seen across the encryption layer:

```
= multipart/encrypted; protocol="application/pgp-encrypted";
  + application/pgp-encrypted [ Version: 1 ]
  + application/octet-stream; name="msg.asc"
    {
      Content-Disposition: inline; filename="msg.asc";
    }
       |
    [ opaque encrypted structure ]
       |
    { minimal headers }
    + multipart/mixed
      + message/rfc822; forwarded="no";
           |
        { protected message headers }
        + multipart/mixed
          + multipart/alternate
            + text/plain
            + text/html
          + application/octet-stream [ attachmet_1 ]
          + application/octet-stream [ attachmet_2 ]
      + application/pgp-keys
```

The header fields of the sub-part of type application/octet-stream SHOULD be modified to ensure that:

- the Content-Type header field's
  - "name" parameter is set to the value "msg.asc", and
  - parameter "forwarded" is set to "no", and
- the Content-Disposition header field value is set to "inline"
  - and the "filename" parameter is set to "msg.asc".

5.7. S/MIME Compatibility

Interoperability and implementation symmetry between PGP/MIME and S/MIME is on the roadmap of pEp.

6. Candidate Header Fields for Header Protection

By default, all headers of the original message SHOULD be wrapped with the original message, with one exception:

- the header field "Bcc" MUST NOT be added to the protected headers.

7. Stub Outside Headers

The outer message requires a minimal set of headers to be in place for being eligible for transport. This includes the "From", "To", "Cc", "Bcc", "Subject" and "Message-ID" header fields. The protocol hereby defined also depends on the "MIME-Version", "Content-Type", "Content-Disposition" and eventually the "Content-Transport-Encoding" header field to be present.

Submission and forwarding based on SMTP carries "from" and "receivers" information out-of-band, so that the "From" and "To" header fields are not strictly necessary. Nevertheless, "From", "Date", and at least one destination header field is mandatory as per [RFC5322]. They SHOULD be conserved for reliability.

The following header fields should contain a verbatim copy of the header fields of the inner message:

- Date
- From
- To
- Cc (*)

- Bcc (*)

The entries with an asterisk mark (*) should only be set if also present in the original message.

Clients which follow pEp standards MUST set the header field value for "Subject" to "=?utf-8?Q?p=E2=89=A1p?=" or "pEp". Clients which do not adhere to all pEp standards should set the header field value of "Subject" to a descriptive stub value. An example used in practice is

- Subject: Encrypted message

The following header fields MUST be initialized with proper values according to the MIME standards:

- Content-Type
- Content-Disposition
- Content-Transport-Encoding (if necessary)

8. Processing Incoming Email under Progressive Header Disclosure

[[TODO]]

8.1. Resolving Conflicting Protected and Unprotected Header Fields

Header field values from the transport message MUST NOT be shown, when displaying the inner message, or the outer message. The inner message MUST carry all relevant header fields necessary for display.

8.2. Processing of Signed-only Email

pEp either engages in a signed-and-encrypted communication or in an unsigned plaintext communication. Inbound signatures attached to plaintext messages are duly verified but cannot enhance the perceived quality of the message in the user interface (while an invalid signature degrades the perception) [I-D.birk-pep].

8.3. Incoming Filter Processing

The Mail User Agent may implement outgoing filtering of mails, which may veto, alter, redirect or replicate the messages. The functionality may be implemented on the mailbox server and be configurable through a well-known protocol, e.g., by means of The Sieve Mail-Filtering Language [RFC5490], or be implemented client-side, or in a combination of both.

A mailbox server, which is required to process the full range of possible filters, is requiring plaintext access to the header fields.

In an end-to-end-encryption context, which pEp enforces by default, upon first reception of the message the mailbox server is limited to see the transport- relevant headers of the outer wrapper message. A pEp client configured to trust the server ("trusted server" setting [I-D.marques-pep-email]) will later download the encrypted message, decrypt it and replace the copy on the server by the decrypted copy.

8.3.1. Staged Filtering of Inbound Messages

Inbound messages are expected to be delivered to the inbox while still being encrypted. At this point in time, server-side filtering can only evaluate the unprotected header fields in the wrapper message.

In an end-to-end-encryption context, which pEp enforces by default, the mailbox server is limited to see the transport-relevant headers of the outer wrapper message only upon first delivery. A pEp client configured to trust the server ("trusted server" setting [I-D.marques-pep-email]) will eventually download the encrypted message, decrypt it locally and replace the copy on the server by the decrypted copy. Server-side message filters SHOULD be able to detect such post-processed messages, and apply the pending filters. The client SHOULD easily reflect the post-filtered messages in the user interface.

8.4. Outgoing Filter Processing

The Mail User Agent may implement outgoing filtering of emails, which may veto, alter or replicate the email. They may also hint the MUA to store a copy in an alternate "Sent" folder.

Filters which veto the sending or do alter the mail or replicate it (e.g., mass-mail generators) SHOULD be completed prior to applying protection, and thus also prior to applying header protection. Redirection to alternate "Sent" folders MUST NOT be decided prior to applying protection, but MUAs MAY abide from this restriction if they implement the "trusted server" option and the option is selected for the specific mailbox server; in this case, MUAs MUST NOT allow to redirect a message to an untrusted server by these rules, to prevent information leakage to the untrusted server.

[[TODO: Mention implicit filter for minimal color-rating for message replication.]]

[[TODO: How to produce key-export-mails manually this way? That is, what about non-pEp-mode?]]

9. Security Considerations

[[TODO]]

10. Implementation Status

10.1. Introduction

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "[...] this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit."

10.2. Current software implementing pEp

The following software implementing the pEp protocols (to varying degrees) already exists:

- pEp for Outlook as add-on for Microsoft Outlook, release [SRC.pepforoutlook]
- pEp for Android (based on a fork of the K9 MUA), release [SRC.pepforandroid]
- Enigmail/pEp as add-on for Mozilla Thunderbird, release [SRC.enigmailpep]
- pEp for iOS (implemented in a new MUA), beta [SRC.pepforios]

pEp for Android, iOS and Outlook are provided by pEp Security, a commercial entity specializing in end-user software implementing pEp while Enigmail/pEp is pursued as community project, supported by the pEp Foundation.

All software is available as Free Software and published also in source form.

11. Acknowledgements

Special thanks go to Krista Bennett and Volker Birk for valuable input to this draft and Hernani Marques for reviewing.

12. References

12.1. Normative References

[I-D.birk-pep]
Marques, H. and B. Hoeneisen, "pretty Easy privacy (pEp): Privacy by Default", draft-birk-pep-03 (work in progress), March 2019.

[I-D.ietf-lamps-rfc5751-bis]
Schaad, J., Ramsdell, B., and S. Turner, "Secure/ Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", draft-ietf-lamps-rfc5751-bis-12 (work in progress), September 2018.

[I-D.marques-pep-email]
Marques, H., "pretty Easy privacy (pEp): Email Formats and Protocols", draft-marques-pep-email-02 (work in progress), October 2018.

[I-D.melnikov-lamps-header-protection]
Melnikov, A., "Considerations for protecting Email header with S/MIME", draft-melnikov-lamps-header-protection-00 (work in progress), October 2018.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3156] Elkins, M., Del Torto, D., Levien, R., and T. Roessler, "MIME Security with OpenPGP", RFC 3156, DOI 10.17487/RFC3156, August 2001, <https://www.rfc-editor.org/info/rfc3156>.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <https://www.rfc-editor.org/info/rfc4880>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[RFC5490] Melnikov, A., "The Sieve Mail-Filtering Language -- Extensions for Checking Mailbox Status and Accessing Mailbox Metadata", RFC 5490, DOI 10.17487/RFC5490, March 2009, <https://www.rfc-editor.org/info/rfc5490>.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010, <https://www.rfc-editor.org/info/rfc5751>.

[RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", STD 76, RFC 6376, DOI 10.17487/RFC6376, September 2011, <https://www.rfc-editor.org/info/rfc6376>.

[RFC7208] Kitterman, S., "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1", RFC 7208, DOI 10.17487/RFC7208, April 2014, <https://www.rfc-editor.org/info/rfc7208>.

[RFC7489] Kucherawy, M., Ed. and E. Zwicky, Ed., "Domain-based Message Authentication, Reporting, and Conformance (DMARC)", RFC 7489, DOI 10.17487/RFC7489, March 2015, <https://www.rfc-editor.org/info/rfc7489>.

12.2. Informative References

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[SRC.enigmailpep] "Source code for Enigmail/pEp", March 2019, <https://enigmail.net/index.php/en/download/source-code>.

[SRC.pepforandroid] "Source code for pEp for Android", March 2019, <https://pep-security.lu/gitlab/android/pep>.

[SRC.pepforios] "Source code for pEp for iOS", March 2019, <https://pep-security.ch/dev/repos/pEp_for_iOS/>.

[SRC.pepforoutlook] "Source code for pEp for Outlook", March 2019, <https://pep-security.lu/dev/repos/pEp_for_Outlook/>.

Appendix A. Document Changelog

[[RFC Editor: This section is to be removed before publication]]

- draft-luck-lamps-pep-header-protection
  - Initial version

Appendix B. Open Issues

[[RFC Editor: This section should be empty and is to be removed before publication.]]

- Align with specification for MIME Content-Type message/partial
  - We probably have issues and overlapping specifications about encoding for nested message/rfc822 entities, specified in [RFC2046]. Further study is needed to find and understand the issues.
- Signed-only protection needs further study
  - pEp only does header protection by applying both signing and encryption. Technically it is also possible to sign, but not encrypt the protected messages. This needs further study.

Authors' Addresses

Claudio Luck
pEp Foundation
Oberer Graben 4
CH-8400 Winterthur
Switzerland

Email: claudio.luck@pep.foundation
URI: https://pep.foundation/

Bernie Hoeneisen
Ucom Standards Track Solutions GmbH
CH-8046 Zuerich
Switzerland

Phone: +41 44 500 52 44
Email: bernie@ietf.hoeneisen.ch (bernhard.hoeneisen AT ucom.ch)
URI: https://ucom.ch/