Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 16, 2019

H. Marques
pEp Foundation
July 15, 2018

# pretty Easy privacy (pEp): Email Formats and Protocols
# draft-marques-pep-email-01

Abstract

The pretty Easy privacy (pEp) propositions for email are based upon already existing email and encryption formats (i.e., PGP/MIME) and designed to allow for easy implementable and interoperable opportunistic encryption: this ranging from key distribution to mechanisms of subject encryption.

The goal of pEp for email is to automatize operations in order to make email encryption usable by a wider range of Internet users, to achieve wide application of confidentiality and privacy practices in the real world.

This document defines basic operations of pEp's approach towards email and two PGP/MIME formats (pEp Email Format 1 and 2) to provide certain security guarantees.

The proposed operations and formats are targeted to Opportunistic Security scenarios and are already implemented in several applications of pretty Easy privacy (pEp).

Status of This Memo

Table of Contents

## 1.  Introduction

This document contains specific propositions to those parts of pretty Easy privacy (pEp) [I-D.birk-pep] that are specific to email. [RFC5322]

All changes required for the pEp propositions on email to work just affect implementers of Mail User Agents (MUAs).

pretty Easy privacy (pEp) for email is a proposition to both, implementers and Internet users, to make end-to-end encryption of emails straightforward.

Whereas Pretty Good Privacy (PGP) and OpenPGP [RFC4880] provide a basis for good encryption, we still miss implementations that also provide a sufficient level of usability for ordinary Internet users.

Two users using pEp-enabled mail clients basically don't need to do anything else than just writing emails.

The following example roughly describes a typical pEp scenario:

1.  Alice - knowing nothing of Bob - just writes an email to Bob: this mail is sent out unencrypted.  However, Alice's public key is automatically attached.

2.  Bob can just reply to Alice and - as he got her public key - is now able to encrypt a message at this point.  Through a color-rating (cf.  [I-D.marques-pep-rating] Bob becomes aware of his message now going out in a secure fashion.

3.  As Alice receives Bob's key, as of now she is also able to send secure messages to Bob.

4.  If Alice and Bob want to prevent man-in-the-middle (MITM) attacks, they can engage in a Handshake [I-D.marques-pep-handshake], comparing their so-called Trustwords [I-D.birk-pep-trustwords] and confirm this process if those match.  After doing so, their identity rating changes to "encrypted and authenticated" [I-D.marques-pep-rating], which (UX-wise) can be signaled, e.g., using a green color rating. This color rating is also applied to messages (in- and outgoing).

This workflow is implemented as running code already in various pEp-enabled software, cf. Section 10.

Note: No propositions are made at this point in time that would require implementers to change the behavior or feature set of email servers. Another Internet-Draft may propose changes to the Simple Mail Transfer Protocol (SMTP) [RFC5321] as to allow for onion routing of email messages in a way metadata can be furtherly protected for communication peers - achievable by message encapsulation. pEp's email message format 2 described below is already prepared for this scenario.

## 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].

- o Handshake: The process when Alice - e.g. in-person or via phone - contacts Bob to verify Trustwords (or by fallback: fingerprints) is called Handshake. [I-D.marques-pep-handshake]

- o Trustwords: A scalar-to-word representation of 16-bit numbers (0 to 65535) to natural language words. When doing a Handshake, peers are shown combined Trustwords of both public keys involved to ease the comparison. [I-D.birk-pep-trustwords]

- o Trust on First Use (TOFU): cf. [RFC7435]

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

## 3. Opportunistic Security with pEp for email

### 3.1. Automatic keypair generation

For every email account a user has in a pEp-enabled Mail User Agent (MUA), a different keypair SHOULD be used by default. If there are no keys whatsoever, RSA-4096 keypairs for OpenPGP encryption [RFC4880] SHOULD be generated automatically for each email account. However, the key length MUST be at least RSA-2048.

If for an identity there's an RSA keypair with less than 2048 bits, new keys MUST be generated.

### 3.2. Key Distribution

By default, public keys MUST always be attached to any outgoing message.

## 4. Encryption of email header fields and interoperability

In pEp, implementers MUST put privacy first: email metadata (i.e., headers) MUST either be omitted or encrypted whenever possible.

In case of email header encryption: implementers of pEp SHOULD be liberal in accepting other approaches to encrypt email headers, but MUST use the strict and interoperable pEp formats for any outgoing communication.

## 5. pEp message formats for email

The pEp message formats 1 and 2 (as described in the following) are email security formats used for sending signed and encrypted emails whenever public key(s) for the recipient(s) exist.

### 5.1. Unencrypted plain text message with public key attached

If for a recipient there's no public key available, a pEp message MUST be sent out in plain text as MIME message version 1, with "Content-Type: multipart/mixed" and the OpenPGP public key attached in ASCII armored format, named "pEpkey.asc".

For a MUA implementer this fulfills two functions:

1. It can be easily detected that the sender is a pEp user.

2. The MUA (if at least OpenPGP-enabled) can enable the receiving user to import the public key to engage in end-to-end encryption with the sender; a MUA implementer can also decide to automatically import the key such that the user can immediately engage in opportunistic encryption.

The plain text messages SHOULD be sent out with the UTF-8 charset Content-Type set.

#### 5.1.1. Example

Please note that in the following examples the "pEpkey.asc" attachments encoded in base64 format are only shown in its first and last line (and otherwise shortened by three points).

```
From: John Doe <jdoe@machine.example>
To: Mary Smith <mary@example.net>
Subject: Test
MIME-Version: 1.0
Content-Type: multipart/mixed;
              boundary="----3YNFBU8B6LV244ZJNQZL12LVUAPGG6"
Content-Transfer-Encoding: 7bit

------3YNFBU8B6LV244ZJNQZL12LVUAPGG6
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain;
charset=UTF-8

Test

------3YNFBU8B6LV244ZJNQZL12LVUAPGG6
Content-Type: application/pgp-keys;
 name="pEpkey.asc"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
 filename="pEpkey.asc";
 size=3813

LS0tLS1CRUdJTiBQR1AgUFVCTElDIEtFWSBCTE9DSy0tLS0tCgptUUlOQkZxNWlkd0JF
...
cHhSUXFhQT09Cj1adlFnCi0tLS0tRU5EIFBHUCBQVUJMSUMgS0VZIEJMT0NLLS0tLS0K

------3YNFBU8B6LV244ZJNQZL12LVUAPGG6--
```

## 5.2. pEp email format version 1

pEp email format 1 is an encrypted and signed PGP/MIME format, which by default ensures:

o  correctly signed messages

o  delivery of public keys (at least and automatically: the sender's public key)

By default, when a public key for a peer is available, pEp-capable MUAs are REQUIRED to send out email messages according to [RFC5322] and in PGP/MIME format [RFC3156] with the informational "Subject:" header field set to "pEp", as follows:

```
Subject: pEp
```

In turn, the intended human-readable subject (in pEp called short message) MUST be moved to the body of the message (in pEp called long

message) and appear as the first line there. pEp implementers are REQUIRED to display the intended "Subject:" field as the real subject line in the respective MUAs to help users to easily grasp the real subject.

Alternatively, the "Subject:" header field can also be set to its UTF-8 variant with "pEp" written with the equivalence symbol instead of an "E":

```
Subject: =?utf-8?Q?p=E2=89=A1p?=
```

Additionally, a header field "X-Pep-Version: 1.0" is added to signal compatibility with pEp email format to pEp-enabled MUAs.

### 5.2.1. Example 1

Example.  Using the well-known example of [RFC5322], an email message sent out with pEp in message format 1 looks like this:

```
      From: John Doe <jdoe@machine.example>
      Sender: Michael Jones <mjones@machine.example>
      To: Mary Smith <mary@example.net>
      Subject: pEp
      Date: Fri, 30 Jun 2018 09:55:06 +0200
      Message-ID: <1234@local.machine.example>
      MIME-Version: 1.0
      Content-Type: multipart/mixed;
                    boundary="----=_NextPart_000_0016_01D0E64A.33EC31B0"
      Content-Language: en-us
      X-Pep-Version: 1.0

      This is a multipart message in MIME format.

      ------=_NextPart_000_0016_01D0E64A.33EC31B0
      Content-Type: text/plain; charset="utf-8"
      Content-Transfer-Encoding: 7bit

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

      ------=_NextPart_000_0060_01D0EAEF.2D54F450
      Content-Type: application/pgp-keys; name="pEp_key.asc"
      Content-Transfer-Encoding: 7bit
      Content-Disposition: attachment; filename="pEp_key.asc"

      -----BEGIN PGP PUBLIC KEY BLOCK-----
      mQINBFQRqIcBEACpsz3mK1zqPdqDlxU6Yws/Xz14LJpszDLlKJckpa7hSc9jfZ4Q
      ...
      Ag7IIk/Gj628hYTdCpNCUc9b1vS6xMAkxJWYgNVwLFS2goikEHCiyzDe
      =MicJ
      -----END PGP PUBLIC KEY BLOCK-----

      ------=_NextPart_000_0060_01D0EAEF.2D54F450--
```

### 5.2.2. Example 2

Using the UTF-8 variant of writing "pEp" with the equivalence symbol, and an additional document attached (an example PDF attachment), an OpenPGP-signed and -encrypted pEp email would look like the following:

```
From: John Doe <jdoe@machine.example>
Sender: Michael Jones <mjones@machine.example>
To: Mary Smith <mary@example.net>
Subject: =?utf-8?Q?p=E2=89=A1p?=
Date: Fri, 30 Jun 2018 09:55:06 +0200
Message-ID: <1234@local.machine.example>
MIME-Version: 1.0
Content-Type: multipart/mixed;
              boundary="----=_NextPart_000_0016_01D0E64A.33EC31B0"
Content-Language: en-us
X-Pep-Version: 1.0

This is a multipart message in MIME format.

------=_NextPart_000_0016_01D0E64A.33EC31B0
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: 7bit

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

------=_NextPart_000_003A_01D10CF6.2DA15150
Content-Type: application/octet-stream; name="example.pdf.pgp"
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="example.pdf.pgp"

-----BEGIN PGP MESSAGE-----
hQIMA/bohV/mG7k7ARAAyy+sdpZYZBhUH/p0gJ+wIlEGTTG2rjLpLuixBrm5Cuj3
...
oAXrQJJgD0F3Ung24Kkundua2gSa9cyeYvUXtA2mbXT7YyN7RdxrMFNfdVFqXZEc
pXqIjL2uKBbyjpS44fc3GmOZNih3bI6q8nl/
=Mvna

------=_NextPart_000_0060_01D0EAEF.2D54F450
Content-Type: application/pgp-keys; name="pEp_key.asc"
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="pEp_key.asc"

-----BEGIN PGP PUBLIC KEY BLOCK-----
mQINBFQRqIcBEACpsz3mK1zqPdqDlxU6Yws/Xz14LJpszDLlKJckpa7hSc9jfZ4Q
...
Ag7IIk/Gj628hYTdCpNCUc9b1vS6xMAkxJWYgNVwLFS2goikEHCiyzDe
=MicJ
```

```
   -----END PGP PUBLIC KEY BLOCK-----

   ------=_NextPart_000_0060_01D0EAEF.2D54F450--
```

## 5.3. pEp email format version 2

pEp email format 2 is a strict PGP/MIME format, which by default ensures:

- correctly signed messages

- delivery of public keys (at least: the sender's public key)

In pEp email format 2 the actual email is encapsulated by an outside multipart/encrypted envelope email (i.e., the actual email is sent like a forwarded message).

Headers of messages (received, to be forwarded etc.) can thus be preserved in the inner message, which is OpenPGP-signed and -encrypted by the application/pgp-encrypted "Content-Type".

In the outer envelope, unnecessary email headers MUST be omitted to the fullest extent.

In contrast to pEp email format 1, the public key and other files attached cannot be seen in the MIME tree.  The only part which can be seen is an application/octet-stream "Content-Type" with name "msg.asc".

### 5.3.1. Example (Outer and Inner Envelope)

A pEp email format 2 message, with the "Subject:" header field set to "pEp" looks like the following (please note that the inner envelope is fully contained in the OpenPGP-signed and -encrypted file named "msg.asc", including possible attachments and with the sender's public key as "pEpkey.asc" attached at the very end):

```
From: John Doe <jdoe@machine.example>
Sender: Michael Jones <mjones@machine.example>
To: Mary Smith <mary@example.net>
Subject: =?utf-8?Q?p=E2=89=A1p?=
Date: Fri, 30 Jun 2018 09:55:06 +0200
Message-ID: <1234@local.machine.example>
MIME-Version: 1.0
Subject: pEp
X-Pep-Version: 2.0
Content-Type: multipart/encrypted;
              boundary="261a304d18692673570d913f7e24b8cb";
              protocol="application/pgp-encrypted"

--261a304d18692673570d913f7e24b8cb
Content-Type: application/pgp-encrypted

Version: 1
--261a304d18692673570d913f7e24b8cb
Content-Type: application/octet-stream
Content-Transfer-Encoding: 7bit
Content-Disposition: inline; filename="msg.asc"

-----BEGIN PGP MESSAGE-----

hQGMAzDKu5MiiyCzAQv9Edg8ulxgxyQfiZRxOpThL0aMFkK7JZH7AJfgdxunLAJk
...
a2jDdzNxotItZk8tWW2h/REdKtRMyXg633DyFLbsIx+cCMnMR1NDChCzvyzUjAw6
XeCGXnY3LB1K
=sdgE
-----END PGP MESSAGE-----

--261a304d18692673570d913f7e24b8cb--
```

The inner envelope in a simple form without further nesting might look like the following, when decrypted:

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="17d3c87b380049a821c764604aaf9272"

--17d3c87b380049a821c764604aaf9272
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: quoted-printable
Content-Disposition: inline; filename="msg.txt"

Subject: The real encrypted subject

Hello, there!

--17d3c87b380049a821c764604aaf9272
Content-Type: application/pgp-keys
Content-Disposition: attachment; filename="pEpkey.asc"

-----BEGIN PGP PUBLIC KEY BLOCK-----
mQGNBFmwE70BDACyR/yQ48QSaQAZyvyUgp7f/4WXxiX1OS9vC/UuewdGLosvl3G+
...
A0KQ6HDwLFuLzneg6Nse4pX0hNWGbLNCouYKdL3vfUHokqp/MTzxyPQlOadDHrDV
H9RC4kMrB/ONGe5yn+u4zjrgq9gWCbdJ43fMoiU3lfMIKy5sZ2NPzh9l
=p5bZ
-----END PGP PUBLIC KEY BLOCK-----
```

It does not only carry the encrypted subject, which pEp implementers are supposed to map (UX-wise) such as to replace the "pEp" subject in the outer envelope, but also the actual message (as inline file named "msg.txt" in case of plain text) as well as the sender's public key.

## 6. Rendering Incoming Messages and Message Rating

pEp-enabled clients MUST NOT blindly render messages. Special care MUST be taken when rendering the pEp email formats, which provide certain guarantees:

| Message Format | Error State | Render | Status Code |
|---|---|---|---|
| PGP/MIME | Unsigned | Yes | DECRYPTED_BUT_UNSIGNED |
| | Signed, no key | Yes | NO_KEY_FOR_SIGNER |
| | Bad signature | No | SIGNATURE_DOES_NOT_MATCH |
| pEp Email 1.0 | Unsigned | No | DECRYPTED_BUT_UNSIGNED |
| | Signed, no key | No | NO_KEY_FOR_SIGNER |
| | Bad signature | No | SIGNATURE_DOES_NOT_MATCH |
| pEp Email 2.0 | Unsigned | No | MODIFICATION_DETECTED |
| | Signed, no key | No | MODIFICATION_DETECTED |
| | Bad signature | No | SIGNATURE_DOES_NOT_MATCH |

For cases where messages appear unsigned, signed without a key or with a bad signature, pEp's privacy rating can be employed to signal issues to a user in an easily understandable manner, cf. [I-D.marques-pep-rating].

[[TODO: This needs more work to be understandable. ]]

## 7. Encryption to Bcc recipients

### 7.1. Algorithm

For encryption of emails that contain Bcc recipients a simple algorithm MAY be used.

Recipients MUST be partitioned into three lists, one for each of three possible outgoing messages:

1. To and Cc recipients (without Bcc recipients)

2. Bcc recipients unable to encrypt

3. Bcc recipients able to encrypt

It's RECOMMENDED that if the original message the user drafted is saved in the user's sent folder, that all recipient fields ("To:", "Cc:", "Bcc:") be preserved.

### 7.1.1. Split To and Cc recipients from Bcc recipients

To and Cc recipients MUST be split from the Bcc recipients.

### 7.1.2. Split Bcc recipients in two groups

Bcc recipients MUST be split in two groups:

o  First group of Bcc recipients who will receive clear text emails.

o  Second group of Bcc recipients who are able to receive encrypted emails.

### 7.1.3. Send one email with only To/Cc recipients

The original email the user drafted SHOULD be sent out with the "Bcc:" field removed.

### 7.1.4. Send one Bcc email for the first Bcc group

For the first Bcc group, a regular email message with only Bcc recipients is sent.

### 7.1.5. Send individual Bcc emails for the second group

For the second group, individual Bcc email messages are sent.

[[TODO: This needs more work to make it better understandable. ]]

## 8. Saving messages

In accordance to the Privacy by Default principle, messages sent or received in encrypted form SHALL be saved with the peer's respective public key.

Messages sent or received in unencrypted form, SHOULD NOT be saved in encrypted form on the mail server: this reflects the Privacy Status the user encountered when sending or receiving the email and thus meets the user's expectations.

Instead, message drafts MUST always be saved with the user's public key.

Other messages sent and received MUST be saved encrypted by default: for most end-user scenarios, the servers users work with, are considered untrusted.

For trusted environments (e.g., in organizations) and to conform to legally binding regulations, pEp implementations MUST provide a "Trusted Server" option.  With the user's explicit consent (opt-in), unencrypted copies of the messages SHALL be held on the mail servers controlled by the organization.  This can also help end-users to archive their emails without needing access to any key material.

## 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:

o  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 Bennet and Volker Birk for the reference implementation on pEp and the ideas leading to this draft.

This work was initially created by pEp Foundation, and will be reviewed and extended with funding by the Internet Society's Beyond the Net Programme on standardizing pEp. [ISOC.bnet]

## 12. References

### 12.1. Normative References

[I-D.birk-pep]
Birk, V., Marques, H., and S. Shelburn, "pretty Easy privacy (pEp): Privacy by Default", draft-birk-pep-02 (work in progress), June 2018.

[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>.

[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>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <https://www.rfc-editor.org/info/rfc7435>.

## 12.2. Informative References

[I-D.birk-pep-trustwords]
Birk, V., Marques, H., and B. Hoeneisen, "IANA Registration of Trustword Lists: Guide, Template and IANA Considerations", draft-birk-pep-trustwords-02 (work in progress), June 2018.

[I-D.marques-pep-handshake]
Marques, H. and B. Hoeneisen, "pretty Easy privacy (pEp): Contact and Channel Authentication through Handshake", draft-marques-pep-handshake-00 (work in progress), June 2018.

[I-D.marques-pep-rating]
Marques, H. and B. Hoeneisen, "pretty Easy privacy (pEp): Mapping of Privacy Rating", draft-marques-pep-rating-00 (work in progress), July 2018.

[ISOC.bnet]
Simao, I., "Beyond the Net. 12 Innovative Projects Selected for Beyond the Net Funding. Implementing Privacy via Mass Encryption: Standardizing pretty Easy privacy's protocols", June 2017, <https://www.internetsociety.org/blog/2017/06/12-innovative-projects-selected-for-beyond-the-net-funding/>.

[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>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <https://www.rfc-editor.org/info/rfc5321>.

[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", July 2018, <https://enigmail.net/index.php/en/download/source-code>.

[SRC.pepforandroid]
"Source code for pEp for Android", July 2018, <https://pep-security.lu/gitlab/android/pep>.

[SRC.pepforios]
"Source code for pEp for iOS", July 2018, <https://pep-security.ch/dev/repos/pEp_for_iOS/>.

[SRC.pepforoutlook]
"Source code for pEp for Outlook", July 2018, <https://pep-security.lu/dev/repos/pEp_for_Outlook/>.

## Appendix A. Document Changelog

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

- draft-marques-pep-email-01:
  * Remove an artefact, fix typos and minor editorial changes; no changes in content
- draft-marques-pep-email-00:
  * Initial version

## Appendix B. Open Issues

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

- Ship better example of pEp Message Format 2
- Elaborate on omitting headers and better explain pEp Message Format 2
- Add notes on EFAIL
- Describe KeyImport to induce the import from secret keys from other devices
- Describe / Reference KeySync (and other sync, through IMAP)
- Add keypair revocation strategy

- Better describe required MIME fields and parameters to set for the pEp email formats

- Create clearer relations to the pEp rating draft (draft-marques-pep-rating), as this plays an important role in how messages are rendered and how they need to be presented (after rating) for a user to have awareness about his privacy status in any given situation.

- Make document more coherent: check with pEp's general draft pieces to fill on both sides and how to reference them vice-versa.

## Author's Address

Hernani Marques
pEp Foundation
Oberer Graben 4
CH-8400 Winterthur
Switzerland

Email: hernani.marques@pep.foundation
URI: https://pep.foundation/