Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 23 November 2025

H. Marques
B. Hoeneisen
pEp Project

22 May 2025

pretty Easy privacy (pEp): Email Formats and Protocols
draft-pep-email-03

Abstract

The proposed pretty Easy privacy (pEp) protocols for email are based upon already existing email and encryption formats (such as PGP/MIME) and designed to allow for easily implementable and interoperable opportunistic encryption. The protocols range from key distribution, secret key synchronization between own devices, to mechanisms of metadata and content protection. The metadata and content protection is achieved by moving the whole message (not only the body part) into the PGP/MIME encrypted part. The proposed pEp Email Formats not only achieve simple forms of metadata protection (like subject encryption), but also allow for sending email messages through a mixnet. Such enhanced forms of metadata protection are explicitly discussed within the scope of this document.

The purpose of pEp for email is to simplify and automate operations in order to make usage of email encryption viable for a wider range of Internet users, with the goal of achieving widespread implementation of data confidentiality and privacy practices in the real world.

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

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-pep-email/.

Discussion of this document takes place on the medup non-WG mailing list (mailto:medup@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/medup/. Subscribe at https://www.ietf.org/mailman/listinfo/medup/.

Source for this draft and an issue tracker can be found at https://codeberg.org/pEp/internet-drafts.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document contains propositions for implementers of Mail User Agents (MUAs) seeking to support pretty Easy privacy (pEp) specifically for email [RFC5322]. All the propositions of [I-D.pep-general] also apply to pEp for email. In this document, requirements are outlined for MUAs wanting to establish interoperability and/or to implement pEp for email.

pEp for email builds upon the cryptographic security services offered by PGP/MIME [RFC3156]. The primary goals of pEp for email are:

(1) Maximization of email privacy for Internet actors deploying and using the pretty Easy privacy approach.

(2) Compatibility with legacy or other automatic email encryption solutions in order to preserve privacy to the greatest extent possible.

Interoperability with S/MIME [RFC8551] is a also goal, but at this time there is no specification or running code that achieves this goal.

Legacy tools and implementations have failed to provide a sufficient level of usability to ordinary Internet users, with the result that end-to-end email encryption is seldom used.

While OpenPGP [RFC9580] using PGP/MIME [RFC3156] offers good encryption -- for message contents at least -- more work is needed to achieve the following three objectives of pretty Easy privacy (pEp):

1. make email encryption as automatic as possible,

2. protect as much metadata as possible, and

3. provide an easy way to authenticate communication partners.

A reference implementation of pEp for email is available for all major platforms and it has been ported to many programming languages (cf. Section 13 for an overview).

1.1. Relationship to Other pEp Documents

This document describes the pEp for email protocols. While it specifies details particularly related to pEp for email, it basically inherits the structure of [I-D.pep-general], which describes the general concepts of pEp on a higher level.

For protocol details, constituent pEp mechanisms which also apply to email can be found in documents like [I-D.pep-handshake]), which shows how trust between any two pEp users can be established, [I-D.pep-rating], which describes the privacy indications that can be helpful for regular Internet users or [I-D.pep-keysync], which outlines pEp's peer-to-peer protocol to synchronize secret key material which belongs to the same account and user across various end-devices.

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Terms

The following terms are defined for the scope of this document:

* pEp Handshake: The process of one User contacting another over an independent channel in order to verify Trustwords (or fingerprints as a fallback). This can be done in-person or through established verbal communication channels, like a phone call. [I-D.pep-handshake]

* Trustwords: A representation of 16-bit natural numbers (0 to 65535) as natural language words: For each natural language a fixed number-to-word map can be defined as convention and registered with IANA. Trustwords are generated from the combined public key fingerprints of a both communication partners. Trustwords are used for verification and establishment of trust (for the respective keys and communication partners). [I-D.pep-trustwords]

* Trust On First Use (TOFU): cf. [RFC7435], which states: "In a protocol, TOFU calls for accepting and storing a public key or credential associated with an asserted Identity, without authenticating that assertion. Subsequent communication that is authenticated using the cached key or credential is secure against an MiTM attack, if such an attack did not succeed during the vulnerable initial communication."

* Man-in-the-middle (MITM) attack: cf. [RFC4949], which states: "A form of active wiretapping attack in which the attacker intercepts and selectively modifies communicated data to masquerade as one or more of the entities involved in a communication association."

Note: Historically, MITM has stood for '_Man_-in-the-middle'. However, to indicate that the entity in the middle is not always a human attacker, MITM can also stand for 'Machine-in-the-middle' or 'Meddler-in-the-middle'.

2. pEp Email Design Principles

In addition to the Protocol's Core Design Principles outlined in [I-D.pep-general], the following sections on design principles are applicable to pEp for email applications.

2.1. Privacy by Default

The pEp formats and protocols aim to maximize privacy. Where privacy goals conflict with security goals, the privacy goals MUST take precedence.

Examples:

* pEp implementers MUST NOT make queries to public key servers by default. The reason for this is to make it more resource-intensive for centralized network actors to learn a user's social graph. This is also problematic security-wise, as centralized cryptographic key subversion at-scale is made easier. Instead, key distribution MUST be handled in-band while communicating with other peers.

* Trust information MUST NOT be attached to the communication partners' public keys. This is metadata which MUST be held locally and separately from the keys. Trust is established between the peers directly (peer-to-peer) and no trust information is held centrally (no support for the Web of Trust): that is, while pEp MUST be able to work with OpenPGP keys which carry trust information, this external trust information MUST NOT be used to signal any trust level to the pEp user.

* pEp-enabled MUAs MUST either engage in a signed-and-encrypted communication or in unsigned plaintext communication. While the signatures attached to plaintext messages can be verified, signed-only messages neither increase security nor privacy, so long as the corresponding public key is not authenticated.

2.2. Data Minimization

Data Minimization includes data sparsity and hiding of all technically concealable information whenever possible.

2.2.1. Metadata Protection

Email metadata (i.e., headers) MUST either be omitted or encrypted whenever possible.

The PGP/MIME specification as described in [RFC3156] provides few facilities for metadata protection: while the email body receives protection, the header section remains unprotected.

However, it is possible to also protect the information contained in the header field values by encapsulating the whole message into a MIME entity to be signed and encrypted.

The S/MIME Message Specification [RFC8551] defines a way to also protect the header section in addition to the content of a message:

> 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.

2.3. Interoperability

Implementers of pEp SHOULD be liberal in accepting non-pEp formats to encrypt email contents and metadata. pEp implementations MUST use the strict and interoperable pEp Email Format 1 (cf. Section 8.2) for any outgoing communication to non-pEp users. For communication between pEp users, more privacy-preserving formats (cf. Section 8) MUST be used. pEp Email Format 2 and newer SHOULD NOT be used between users who are not recognized as pEp users (cf. Section 9.3), because for non-pEp users those formats are likely to produce unwanted visual artifacts.

2.4. End-to-End

Several legacy email systems depend on centralized infrastructures, i.e. typical webmail / HTML emails systems. This is in violation of the pEp end-2-end principle. Encryption and decryption of messages MUST be executed on a user’s end-device and MUST NOT depend on any third-party network infrastructure (i.e., any infrastructure outside a user’s direct control).

2.5. Peer-to-Peer

All relevant pEp mechanisms and state information about other peers MUST be held locally, on a peer's end-device. There MUST NOT be any reliance on an email server or even a centralized network component to hold relevant information for peers to be able to communicate or to authenticate themselves. Email servers (like, SMTP or IMAP) are only used as transport infrastructure for messages, but MUST not be relied upon to hold actual state between peers.

2.6. User Interaction

pEp aims to limit user interactions to a minimum. The only additional step an email user normally needs to perform is the Trustword comparison (cf. Section 5), if he opts to ensure the communication partner is trusted. In special situations (e.g. compromised keys) further user interaction may be needed.

3. pEp Email Identity System

In pEp for email, a user is a person or group who can have one or more identities, each represented by an email address. Every identity has an own key attached to it. An email address can also be an alias for an already existing identity, in which case the same key is attached to it.

All information about communication partners, like identities, keys and aliases MUST be held on a user's end-device as state information. This SHOULD be done using a structured format, to facilitate the synchronization of state information across various devices, taking into account multi-device scenarios, which are common today.

In pEp's reference implementation (cf. Section 13), keys are held using the key store of the cryptographic library, while peer-specific state information, including trust information, is held in a simple relational database.

3.1. User

A user in pEp for email is a specific person or group. A user has at least one identity, but can have more.

3.2. Address

In pEp for email, the SMTP address (e.g., mailto:alice@example.org) constitutes the network address.

3.3. Key

For now, a key in pEp for email is an OpenPGP key. Each identity has a default key attached for that identity. This is the public key to be used to encrypt communications to it.

3.4. Identity

An identity in pEp for email is represented by an email URI, like mailto:alice@example.org. Identities are represented by email address URIs because a user may have multiple URIs. For example, if Alice uses mailto:alice@example.org for private purposes, but also wishes to have a public address, she may create another email address, such as anonymous@example.com. Because this is a new URI (mailto:anonymous@example.com), it is considered a new identity for Alice.

By default, pEp-enabled MUAs MUST generate a new key pair during new account configuration, so that a user’s respective identities are not correlated to each other. However, if Alice wants her URIs to be handled as a single identity with one key, she may configure her respective identities as aliases.

For other email URIs pointing to the same identity, see the alias (cf. Section 3.5) concept.

3.5. Alias

Aliases share the same key and identity, e.g., the same key might be used for mailto:alice@example.org as well as for mailto:alice@example.net. That is, both addresses refer to the same identity.

4. Key Management

4.1. Key Generation

A pEp-enabled Mail User Agent MUST consider every email account as a new identity: for each identity, a different key pair MUST be created automatically if no key material of sufficient length is available. By default, RSA-4096 key pairs for OpenPGP encryption [RFC9580] SHOULD be generated automatically for each email account. However, the key length MUST be at least 2048 bits. Elliptic curve keys with at least 256 bits MUST be supported, but SHOULD NOT yet be generated and announced by default for interoperability reasons.

If for an identity there’s an RSA key pair with less than 2048 bits, new keys MUST be generated.

4.2. Private Keys

4.2.1. Storage

[I-D.pep-general] specifies the requirements regarding storage of private keys.

4.2.2. Passphrase

[I-D.pep-general] specifies the requirements regarding passphrases for private keys.

4.2.3. Private Key Export / Import

4.3. Public Key Distribution

By default, public keys MUST always be attached to any outgoing message as described in Section 8. If this is undesired, Passive Mode (cf. Section 7.1) can be activated.

4.4. Key Reset

The specification for Key Reset can be found in [I-D.pep-keyreset] (and [I-D.pep-general]).

5. Trust Management

The following example roughly describes a pEp email scenario with a typical initial message flow to demonstrate key exchange and basic trust management:

1. Alice -- knowing nothing of Bob -- sends an email to Bob. As Alice has no public key from Bob, this email is sent out unencrypted. However, Alice’s public key is automatically attached.

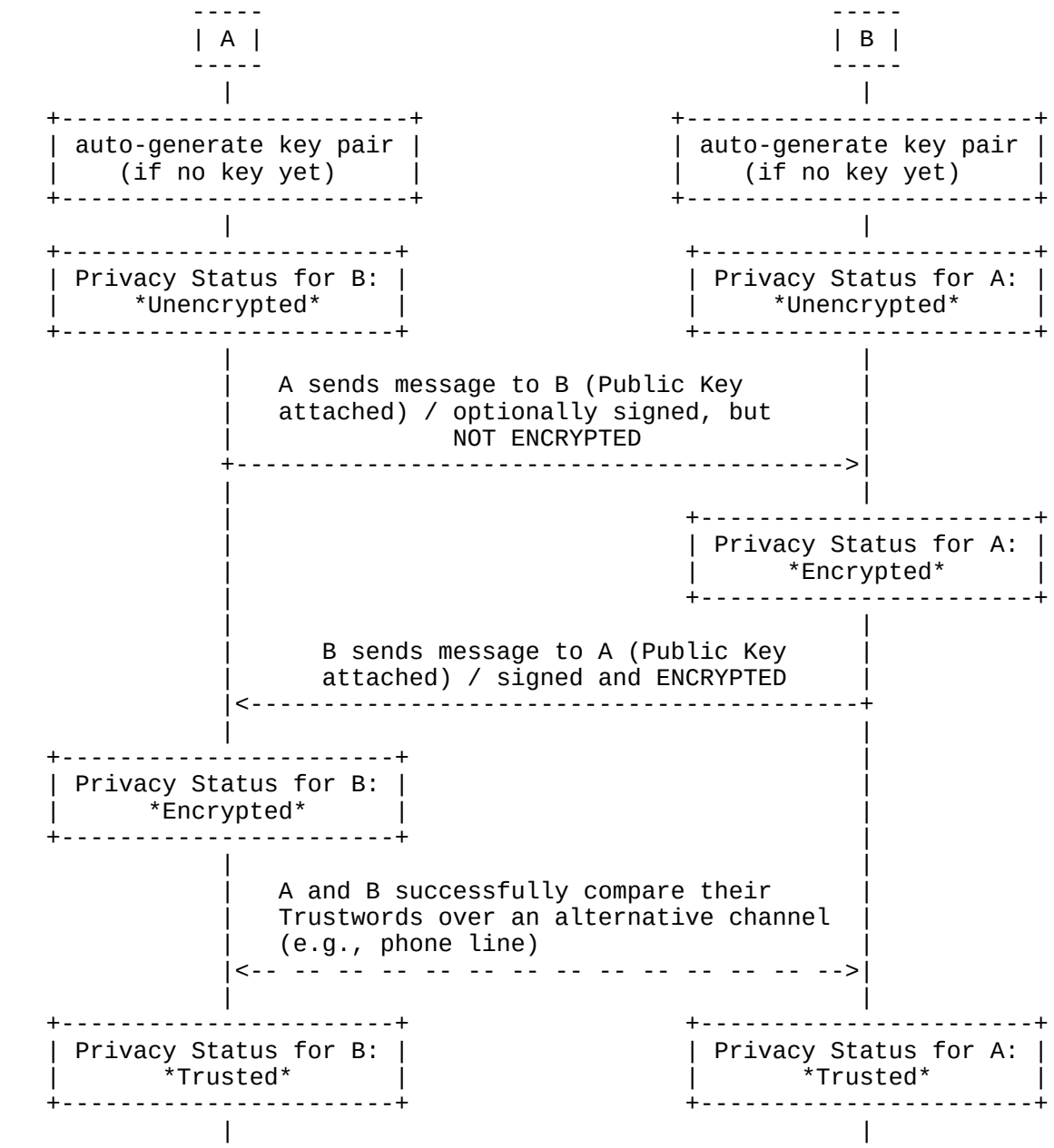
2. Bob can just reply to Alice and -- as he received her public key -- his MUA is now able to encrypt the message. At this point, the rating for Alice changes to "encrypted" in Bob’s MUA, which (UX-wise) can be displayed using yellow color (cf. Section 5.2).

3. Alice receives Bob’s key. As of now Alice is also able to send secure emails to Bob. The rating for Bob changes to "encrypted" (with yellow color) in Alice’s MUA (cf. Section 5.2).

4. If Alice and Bob want to prevent man-in-the-middle (MITM) attacks, they can engage in a pEp Handshake comparing their so-called Trustwords (cf. Section 5.3) and confirm this process if

those match. After doing so, their identity rating changes to "encrypted and authenticated" (cf. Section 5.2), which (UX-wise) can be displayed using a green color.

As color code changes for an identity, this is also reflected to future messages to/from this identity. Past messages, however, MUST NOT be altered.



5.1. Privacy Status

The Privacy Status is further specified in [I-D.pep-rating] (and [I-D.pep-general]).

5.2. Trust Rating

Trust Rating is further specified in [I-D.pep-rating] (and [I-D.pep-general]).

5.3. Handshake

The Handshake is further specified in [I-D.pep-handshake] (and [I-D.pep-general]).

6. Synchronization

pEp offers a mechanism to group different devices to share state between them. How to establish such a sync group is further specified in [I-D.pep-keysync] (and [I-D.pep-general]).

6.1. Private Key Synchronization

The mechanism for synchronizing Private Keys among different devices is further specified in [I-D.pep-keysync] (and [I-D.pep-general]).

7. Options in pEp

7.1. Option "Passive Mode"

The option "Passive Mode" is outlined in [I-D.pep-general].

In email, Passive Mode primarily exists as an option to avoid potential usability issues in certain environments where Internet users might get confused by the exposure of public keys in email attachments. This concerns mainly clients unaware of encryption. In principle, however, this "problem" can be mitigated either by training or by MUA implementers displaying public key material in a more symbolic way or even importing it automatically and then hiding this attachment altogether (as pEp implementers are supposed to do, such that regular Internet users do not have to bother about keys).

Passive Mode has a negative impact on privacy: additional unencrypted message exchanges are needed until pEp’s by-default encryption can take place.

Passive Mode MUST only affect unencrypted communications and MUST be inactive by default. By opting in to Passive Mode, the sender’s public key MUST NOT be attached when sending out unsecure emails. On the other hand, Passive Mode is without any effect when pEp is able to send out an encrypted message, because the necessary encryption key(s) are available.

In this situation, opportunistic by-default encryption MUST take place: there, the sender’s public key is attached in encrypted form as constituent part of one of pEp’s PGP/MIME-based email format described in Section 8.

Additionally, Passive Mode MUST be without effect, if a receiver learns that an MUA is actually pEp-capable, even if the sender involved is in Passive Mode, too: this MUST be recognized by the "X-pEp-Version" header field, as the only clear indicator to detect pEp users. That means that a pEp-enabled MUA is REQUIRED to attach its corresponding public key to another pEp user in any case, such that they can engage in opportunistic encryption.

7.2. Option "Disable Protection"

The option "Disable Protection" is outlined in [I-D.pep-general].

This is an opt-in mechanism to enforce that messages go out unprotected. Even if encryption keys for recipient(s) are available, this option MUST enforce that messages are sent in the Section 8.1 format.

7.3. Option "Extra Keys"

The option "Extra Keys" is described in [I-D.pep-general].

In email this is in particular useful in enterprise setups, e.g., for legal reasons or email escrow purposes.

7.4. Option "Blacklist Keys"

The option "Blacklist Keys" is described in [I-D.pep-general].

7.5. Option "Trusted Server"

The option "Trusted Server" is described in [I-D.pep-general].

8. pEp Email Data Formats

The pEp Email Formats versions 1 and 2 are restricted MIME-based email formats, which ensure messages to be signed and encrypted. In accordance with pEp’s privacy (and not security) focus, signed-only messages MUST NOT be produced (cf. Section 2.1). pEp-enabled clients MUST be able to render all pEp Email Formats properly: for outgoing communications, the most privacy-preserving format available is to be used, taking interoperability (cf. Section 2.3) into account.

pEp Email Format versions 2 (and higher) are normally understood by pEp implementations only. For backward compatibility reasons pEp Email Format 1 has been defined (cf. Section 8.2). If the peer has not indicated to support pEp, but a trustworthy public key is available according to the local database, pEp Email Format 1.1 (cf. Section 8.2) SHOULD be applied.

In case no trustworthy encryption key is available, an unencrypted, unsigned MIME email is sent out. As in all pEp formats, also this (unprotected) message MUST contain the sender's public key, unless Passive Mode (cf. Section 7.1) is active.

All pEp Email Formats include a "pEpkey.asc" file attachment holding the sender's OpenPGP public key in ASCII-armored format, which is suitable for manual key import by non-pEp users. Thus, a user of any OpenPGP-enabled MUA is able to manually import the public key and engage in end-to-end encryption with the pEp sender. MUA implementers of PGP-capable email clients, even when not fully supporting pEp's protocols, are encouraged to automatically import the key such that the user can immediately engage in opportunistic encryption.

In pEp's reference implementation the subject is set to "pEp". (Note that earlier versions used the UTF-8 representation of an alternative stylized form "p≡p".) However, the subject's value of the outer message MUST be ignored. Therefore, the subject can be set to any value (e.g., "[...]" as used in other implementations for encrypted email).

To determine whether or not a peer supports pEp, pEp implementations SHOULD append a Header Field "X-Pep-Version" to the OUTER message containing the sender's highest supported pEp protocol version to every outgoing email, e.g., "X-Pep-Version: 2.1". The pEp protocol version determines, which pEp Email Format is understood by the sender.

By appending a "X-Pep-Version" Header Field, a pEp-enabled MUA declares its capability to receive and render more privacy-preserving formats. Upgrading both sides to the highest pEp version allows pEp-enabled MUAs for best possible protection of metadata.

8.1. Unencrypted pEp Email Format

The unencrypted pEp Email Format by default ensures the delivery of the sender's public key as an attachment. This is the format to be used when unencrypted messages are sent out.

The sender’s public key is added as application/pgp-keys MIME entity to the end of an existing multipart/mixed MIME entity. If the email contains no such multipart/mixed MIME entity, a multipart/mixed MIME entity is inserted.

The MIME structure of an example email containing an alternative text/html plus attachment looks as follows:

```
└─┬ multipart/mixed 6229 bytes
  ├─┬ multipart/alternative 1033 bytes
  │ ├── text/plain 293 bytes
  │ └── text/html 490 bytes
  ├── text/markdown attachment [pef-0.mkd] 800 bytes
  └── application/pgp-keys attachment [sender_key.asc] 3022 bytes
```

In the following how a simple plaintext example email looks:

```
From: Alice <alice@example.org>
To: Bob <bob@example.org>
Date: Tue, 31 Dec 2019 05:05:05 +0200
X-pEp-Version: 2.1
MIME-Version: 1.0
Subject: Saying Hello
Content-Type: multipart/mixed; boundary="boundary"

--boundary
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: quoted-printable

Hello Bob

If you reply to this email using a pEp-enabled client, I will
be able to send you that sensitive material I talked to you
about.

Have a good day!

Alice

--
Sent with pEp for Android.

--boundary
Content-Type: application/pgp-keys; name="pEpkey.asc"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="pEpkey.asc"; size=2639

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

[...]

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

--boundary--
```

8.2. pEp Email Format 1

pEp Email Format 1 (PEF-1) is an encrypted and signed MIME format, which by default ensures:

* a signed and encrypted message, with Subject header field encryption
* delivery of the sender’s public key

On-the-wire PEF-1 contains a "multipart/encrypted" (cf. [RFC1847] / [RFC3156]) MIME node with an OpenPGP encrypted and signed filename "msg.asc", and "Content-Disposition: inline" attribute.

As with the unencrypted pEp Email Format (cf. Section 8.1), PEF-1 adds the sender's public key as application/pgp-keys MIME entity to the end of an existing multipart/mixed MIME entity. If the email contains no such multipart/mixed MIME entity, a multipart/mixed MIME entity is inserted.

The PEF-1 MIME structure of an example email containing an alternative text/html plus attachment looks as follows:

```
└─┬ multipart/encrypted 14143 bytes
  ├── application/pgp-encrypted 10 bytes
  └── application/octet-stream inline [msg.asc] 11534 bytes
    (decrypts to)
    └─┬ multipart/mixed 7329 bytes
      ├─┬ multipart/alternative 1079 bytes
      │ ├── text/plain 324 bytes
      │ └── text/html 510 bytes
      ├── text/markdown attachment [pef-1.mkd] 540 bytes
      └── application/pgp-keys attachment [sender_key.asc] 5354 bytes
```

With PEF-1 the only header field that can be protected is the Subject. To achieve this protection, the real subject value is added to the top of the content section of the very first MIME entity with media type "text/plain" and that is encrypted (embedded subject), e.g.:

```
Subject: Credentials
```

For example, an email with "Subject: Credentials" becomes "Subject: pEp" on-the-wire (after encryption has taken place), with the true subject appended as embedded subject to the top of MIME entity contain with media type "text/plain"

Thus, legacy clients unaware of or unable to parse pEp's PEF-1 subject encryption, still display the actual subject content (in the above example: "Subject: Credentials") to the user, as part of the message body. MUAs implementing pEp MUST render the an embedded subject to the user, i.e. replace whatever was in the Subject Header Field with the content of the embedded subject.

Note that the embedded subject is case insensitive (as Header Fields in email), so that also embedded subjects starting with "subject:" or "SUBJECT:" are to be rendered.

Please note that subject encryption MAY be disabled (cf. Section 7.2). This may be useful for sending messages between pEp- and non-pEp clients, but sacrifices some privacy over usability by avoiding artefacts for non-pEp recipients.

PEF-1 is also considered pEp’s compatibility format towards non-pEp clients.

In the following how a simple PEF-1 example email looks:

```
From: Alice <alice@example.org>
To: Bob <bob@example.org>
Date: Wed, 1 Jan 2020 23:23:23 +0200
X-Pep-Version: 2.1
MIME-Version: 1.0
Subject: pEp
Content-Type: multipart/encrypted; boundary="boundary1";
 protocol="application/pgp-encrypted"

--boundary1
Content-Type: application/pgp-encrypted

Version: 1

--boundary1
Content-Type: application/octet-stream
Content-Transfer-Encoding: 7bit
Content-Disposition: inline; filename="msg.asc"

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

[...]

-----END PGP MESSAGE-----

--boundary--
```

Decrypting the enclosed "msg.msc" part yields the following:

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="boundary2"
--boundary2
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: quoted-printable
Content-Disposition: inline; filename="msg.txt"

Subject: Credentials

Dear Bob

Please use "bob" with the following password to access the wiki site:

correcthorsebatterystaple

Please reach out if there are any issues and have a good day!

Alice

--boundary2
Content-Type: application/pgp-keys
Content-Disposition: attachment; filename="pEpkey.asc"

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

[...]

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

--boundary2--
```

Notes:

* The user-intended subject value is encrypted in the first "text/plain" MIME entity under the "multipart/mixed" MIME node.

* The "X-Pep-Version" Header Field is set to "2.1", as this is an example of of a pEp Client sending the compatibility format towards a non-pEp client.

8.3. pEp Email Format 2

pep Email Format 2 (PEF-2) has been designed to improve privacy and reduce the likelihood for programming mistakes. It is also suitable for mixnet, i.e. routing email through the use of onion routing or mix networks, e.g. [pEp.mixnet].

As PEF-1, on-the-wire PEF-2 contains a "multipart/encrypted" (cf. [RFC1847] / [RFC3156]) MIME node with an OpenPGP encrypted and signed filename "msg.asc", and "Content-Disposition: inline" attribute.

pEp Email Format 2 (PEF-2) uses a pEp-specific MIME structure, i.e a multipart/mixed MIME entity containing 3 MIME entities:

1. A text/plain MIME entity used to inform about the nature of this format (useful in case this ends up at a non-pEp client)

2. A message/rfc822 MIME entity containing the original (aka inner) message

3. An application/pgp-keys MIME entity containing the sender's public key

The PEF-2 MIME structure of an example email containing an alternative text/html plus attachment looks as follows:

```
└─┬ multipart/encrypted 15297 bytes
  ├── application/pgp-encrypted 10 bytes
  └── application/octet-stream inline [msg.asc] 13133 bytes
    (decrypts to)
    └─┬ multipart/mixed 8330 bytes
      ├── text/plain 263 bytes
      ├─┬ message/rfc822 2332 bytes
      │ └─┬ multipart/mixed 2286 bytes
      │   ├─┬ multipart/alternative 1115 bytes
      │   │ ├── text/plain 296 bytes
      │   │ └── text/html 524 bytes
      │   └── text/markdown attachment [pef-2.mkd] 554 bytes
      └── application/pgp-keys attachment [sender_key.asc] 5354 bytes
```

PEF-2 introduces further pEp-specific header fields to the inner message, which help to determine the behavior between pEp users.

In normal interpersonal messaging those additional header fields are:

1. X-pEp-Version: The version of the pEp protocol the sending client is using, e.g., "2.1"

2. X-pEp-Wrapped-Message-Info: Set to "INNER" states that the message carrying this is to be considered the most inner message containing the original email (this is particularly relevant for mixnet or other scenarios of nested messaging; cf. [pEp.mixnet]), and

3. X-pEp-Sender-FPR: The value set to sender’s full 160-bit public key fingerprint (e.g., "1234567890ABCDEF1234567890ABCDEF12345678")

A caveat of PEF-2 is that message rendering varies considerably across different MUAs. This is relevant as it might happen that a non-pEp MUA encounters a PEF-2 message (e.g., if a pEp-enabled client was used in the past). No standard is currently available which enables MUAs to reliably determine whenever a nested "message/rfc822" MIME entity is meant to render the contained email message, or if it was effectively intended to be forwarded as an attachment, where a user needs to click on in order to see its content. To help unaware MUAs, a Content-Type header field parameter with name "forwarded" as per [I-D.melnikov-iana-reg-forwarded] is added to the Content-Type header field. MUAs can use this to distinguish between a forwarded message and a nested message (i.e., using "forwarded=no").

As with PEF-1.0, if the receiving side is not a known pEp-enabled MUA, but there is a trustworthy public key available, PEF-1 (cf. Section 8.2) MUST be used to send the email.

In the following how a simple PEF-2 example email looks:

```
From: Alice <alice@example.org>
To: Bob <bob@example.org>
Date: Wed, 1 Jan 2020 23:23:23 +0200
X-pEp-Version: 2.1
MIME-Version: 1.0
Subject: pEp
Content-Type: multipart/encrypted; boundary="boundary1";
              protocol="application/pgp-encrypted"

--boundary1
Content-Type: application/pgp-encrypted

Version: 1

--boundary1
Content-Type: application/octet-stream
Content-Transfer-Encoding: 7bit
Content-Disposition: inline; filename="msg.asc"

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

[...]

-----END PGP MESSAGE-----

--boundary1--
```

Unwrapping the "multipart/encrypted" MIME node, yields this:

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="boundary2"

--boundary2
Content-Type: text/plain; charset="utf-8"
Content-Disposition: inline; filename="msg.txt"

This message was encrypted with pEp (https://pep.software). If you
are seeing this message, your client does not support raising message
attachments. Please click on the message attachment to view it,
or better yet, consider using pEp!

--boundary2
Content-Type: message/rfc822; forwarded="no"

Message-ID: <pEp.1234>
Date: Wed, 1 Jan 2020 23:23:23 +0200
Subject: Credentials
X-pEp-Version: 2.1
```

```
X-pEp-Wrapped-Message-Info: INNER
X-pEp-Sender-FPR: 44A440451DD3C88744AF3558AA93542C17905D1F
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="boundary3"

--boundary3
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: quoted-printable

Dear Bob

Please use "bob" with the following password to access the wiki site:

correcthorsebatterystaple

Please reach out if there are any issues and have a good day!

Alice

--boundary3--

--boundary2

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

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

[...]

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

--boundary2--
```

9. Email Processing

9.1. Identifiying a pEp Email Format 1 (PEF-1) Message

Messages composed with pEp Email Format 1.0 (cf. Section 8.2) can be identified as follows:

* The message’s outer structure is the "confidentiality, integrity, and authenticity all together" variant of the PGP/MIME Encryption Cryptographic Layer as described in Section 4.1.2.2 of [I-D.ietf-lamps-e2e-mail-guidance].

* The ciphertext in this structure decrypts into the following MIME structure:

```
W └─┬ multipart/mixed
X   ├── [Inner Message]
Y   ├── [attachment(s) of Inner Message (optional)]
Z   └── application/pgp-keys attachment [sender_key.asc]
```

That is, the decrypted structure can be described as:

* Part W (the top-level part of the decrypted ciphertext) is Content-Type: multipart/mixed, and it has at least two child nodes (i.e., Parts X and Z):
  - Part X (first child node) contains the Inner Message
  - Part Y (optional child node(s)) contain(s) attachment(s) of Inner Message
  - Part Z (last child node) is Content-Type: application/pgp-keys

And the decrypted structure also has the following properties:

* Part Z comprises an attachment containing the sender key with filename sender_key.asc

Note: Part Y is optional, i.e., if the Inner Message has no attachment, part Y is not present. Depending on the number of attachments, part Y may have one or more child node instance(s).

9.2. Identifiying a pEp Email Format 2 (PEF-2) Message

Messages composed with pEp Email Format 2.1 (cf. Section 8.3) can be identified as follows:

* The message’s outer structure is the "confidentiality, integrity, and authenticity all together" variant of the PGP/MIME Encryption Cryptographic Layer as described in Section 4.1.2.2 of [I-D.ietf-lamps-e2e-mail-guidance].

* The ciphertext in this structure decrypts into the following MIME structure:

```
A └─┬ multipart/mixed
B   ├── text/plain
C   ├─┬ message/rfc822; forwarded="no"
D   │ └── [Inner Message]
E   └── application/pgp-keys attachment [sender_key.asc]
```

That is, the decrypted structure can be described as:

* Part A (the top-level part of the decrypted ciphertext) is Content-Type: multipart/mixed, and it has exactly three child nodes:
  - Part B is Content-Type: text/plain
  - Part C is Content-Type: message/rfc822 (whose immediate child part D contains the Inner Message)
  - Part E is Content-Type: application/pgp-keys

And the decrypted structure also has the following properties:

* Part C has a Content-Type parameter forwarded="no"
* Part D has the following three Header Fields:
  - X-pEp-Version:, with a value of 2.1 or higher
  - X-pEp-Wrapped-Message-Info:, with a value of INNER
  - X-pEp-Sender-FPR:, with a value of 40 hexadecimal characters, which is the version 4 OpenPGP fingerprint of the sender’s OpenPGP certificate (see [RFC9580])
* Part E comprises an attachment containing the sender key with filename sender_key.asc

9.3. Protocol Negotiation for Format Selection

To be able to decide which Email Format to generate, the pEp-enabled MUA REQUIRES to record state on a per-identity basis. Once a "X-pEp-Version" header field is discovered, the user MUST be recorded as a pEp user and the corresponding pEp protocol version it supports (according to the highest value of the "X-pEp-Version" header field encountered).

9.4. Saving Messages

In accordance with the Privacy by Default principle, messages sent or received in encrypted form MUST be saved with the identity’s respective public key.

Messages sent or received in unencrypted form, SHOULD NOT be saved in encrypted form on the email 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 identity'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 archiving regulations, pEp implementations MUST provide a "Trusted Server" option. With the user's explicit consent (opt-in), unencrypted copies of the Messages MUST be held on the mail servers controlled by the organization. (Cf. Section 7.5.)

10. Interoperability with other Email Systems

pEp aims to be interoperable with existing email applications designed to enable privacy, e.g., OpenPGP [RFC9580] and S/MIME [RFC8551].

11. Security Considerations

First of all, the Security Considerations described in [I-D.pep-general], also apply to this document.

11.1. Security Gap in Exchange of Very First Message

As email communications constitute an offline type of communication, the very first message cannot be securely protected and tampering is possible. Through contact verification as described in [I-D.pep-handshake], however, it is at least possible to detect attacks like MITM afterwards.

11.2. Metadata Surveillance

In pEp for email, full metadata protection is not (yet) possible in transit or if the involved email servers are being inspected.

12. IANA Considerations

This document has no actions for IANA.

13. Implementation Status

13.1. 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."

13.2. Current Software Implementations of pEp

The following applications implementating the pEp protocols already exist:

* pEp for Thunderbird as an add-on for Thunderbird, release [SRC.pepforthunderbird]

* pEp for Outlook as add-on for Microsoft Outlook

* pEp for iOS (implemented in a new MUA)

* pEp for Android (based on a fork of the K9 MUA)

All these applications are using the pEp Engine reference implementation [SRC.pepcore].

Note: The former community project Enigmail/pEp as add-on for Thunderbird was discontinued and replaced by pEp’s own add-on for Thunderbird [SRC.pepforthunderbird] in 2021.

pEp for Android, iOS, Outlook and Thunderbird were provided by pEp Security, a discontinued commercial entity specializing in end-user pEp implementations.

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

14. Acknowledgments

Special thanks go to Krista Bennett and Volker Birk for the reference implementation on pEp and the ideas leading to this draft.

The author would like to thank the following people who provided substantial contributions, helpful comments or suggestions for this document: Berna Alp, Kelly Bristol, Claudio Luck, Daniel Kahn Gillmor, Linda Carmen Schmid, Luca Saiu, Lars Rohwedder, and Nana Karlstetter.

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

15. References

15.1. Normative References

[I-D.melnikov-iana-reg-forwarded]
Melnikov, A. and B. Hoeneisen, "IANA Registration of Content-Type Header Field Parameter ’forwarded’", Work in Progress, Internet-Draft, draft-melnikov-iana-reg-forwarded-00, 4 November 2019, <https://datatracker.ietf.org/doc/html/draft-melnikov-iana-reg-forwarded-00>.

[I-D.pep-general]
Birk, V., Marques, H., and B. Hoeneisen, "pretty Easy privacy (pEp): Privacy by Default", Work in Progress, Internet-Draft, draft-pep-general-03, 22 May 2025, <https://datatracker.ietf.org/doc/html/draft-pep-general-03>.

[I-D.pep-handshake]
Marques, H. and B. Hoeneisen, "pretty Easy privacy (pEp): Contact and Channel Authentication through Handshake", Work in Progress, Internet-Draft, draft-pep-handshake-00, 16 December 2022, <https://datatracker.ietf.org/doc/html/draft-pep-handshake-00>.

[I-D.pep-keyreset]
Hoeneisen, B., "pretty Easy privacy (pEp): Key Reset", Work in Progress, Internet-Draft, draft-pep-keyreset-00, 15 December 2022, <https://datatracker.ietf.org/doc/html/draft-pep-keyreset-00>.

[I-D.pep-rating]
Marques, H. and B. Hoeneisen, "pretty Easy privacy (pEp): Mapping of Privacy Rating", Work in Progress, Internet-Draft, draft-pep-rating-00, 16 December 2022, <https://datatracker.ietf.org/doc/html/draft-pep-rating-00>.

[RFC1847] Galvin, J., Murphy, S., Crocker, S., and N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, DOI 10.17487/RFC1847, October 1995, <https://www.rfc-editor.org/rfc/rfc1847>.

[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/rfc/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/rfc/rfc3156>.

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

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/rfc/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/rfc/rfc7435>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC9580] Wouters, P., Ed., Huigens, D., Winter, J., and Y. Niibe, "OpenPGP", RFC 9580, DOI 10.17487/RFC9580, July 2024, <https://www.rfc-editor.org/rfc/rfc9580>.

15.2. Informative References

[I-D.ietf-lamps-e2e-mail-guidance]
Gillmor, D. K., Hoeneisen, B., and A. Melnikov, "Guidance on End-to-End E-mail Security", Work in Progress, Internet-Draft, draft-ietf-lamps-e2e-mail-guidance-17, 8 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-e2e-mail-guidance-17>.

[I-D.pep-keysync]
Birk, V., Hoeneisen, B., and H. Marques, "pretty Easy privacy (pEp): Key Synchronization Protocol (KeySync)", Work in Progress, Internet-Draft, draft-pep-keysync-03, 6 June 2023, <https://datatracker.ietf.org/doc/html/draft-pep-keysync-03>.

[I-D.pep-trustwords]
Hoeneisen, B. and H. Marques, "IANA Registration of Trustword Lists: Guide, Template and IANA Considerations", Work in Progress, Internet-Draft, draft-pep-trustwords-01, 23 December 2022, <https://datatracker.ietf.org/doc/html/draft-pep-trustwords-01>.

[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/>.

[pEp.mixnet]
pEp Project, "pEp's Mixmailer", May 2025, <https://codeberg.org/pEp/mixmailer>.

[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/rfc/rfc7942>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/rfc/rfc8551>.

[SRC.pepcore]
"Core source code and reference implementation of pEp (engine)", May 2025, <https://codeberg.org/pEp/pEpEngine>.

[SRC.pepforthunderbird]
"Source code for pEp for Thunderbird", May 2025,
<https://codeberg.org/pEp/pEpForThunderbird>.

Appendix A. Document Changelog

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

* draft-pep-email-03:
  - Add Section Identifiying a pEp Email Format 2 (PEF-2) Message
  - Add Section Identifiying a pEp Email Format 1 (PEF-1) Message
  - Update Implementations Status
  - Update References
* draft-pep-email-02:
  - Reorganized chapter structure to align with [I-D.pep-general]
  - Rewrite of pEp Email Formats section
  - Major rewrite and resolving TODO’s
  - Add Security Considerations
* draft-pep-email-01:
  - Minor language improvements
* draft-pep-email-00:
  - Major restructure of the document
  - Major fixes in the description of the various message formats
  - Add many open questions and comments inline (TODO)
  - Add IANA Considerations section
  - Change authors and acknowledgment section
  - Add internal references
  - Describe Passive Mode

- Better explanation on how this document relates to other pEp documents

* draft-marques-pep-email-02:

- Add illustrations

- Minor fixes

- Add longer list of Open Issues (mainly by Bernie Hoeneisen)

* 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]]

* Explain Key Mapping (between OpenPGP and S/MIME)

Authors' Addresses

Hernani Marques
pEp Project
Oberer Graben 4
CH-8400 Winterthur
Switzerland
Email: hernani.marques@pep-project.org
URI: https://pep-project.org/

Bernie Hoeneisen
pEp Project
Oberer Graben 4
CH-8400 Winterthur
Switzerland
Email: bernie.hoeneisen@pep-project.org
URI: https://pep-project.org/