SML Use Cases for Opportunistic Email Encryption in pEp

SML @ IETF-116, Tue 28 March 2023

Hernâni Marques / Bernie Hoeneisen

Privacy by Default.
Background

- pEp aims to make text communications (i.e., email, chat, ...) **private by default**
- pEp systems must not depend on centralized elements
- “Good” tools for privacy (using, e.g., OpenPGP) already exist

**However:**
- Most users are unable to use existing encryption tools such as GnuPG (properly)
- Need to fix this usability challenge by automation
- This automation requires “Structured Emails”
Use Case 1: Attached Public Key

- Public key is attached to pEp email messages (by default)
- Properties (of this type) of Structured Email
  - Key to be automatically processed by receiving MUA
- Receiving users may get confused by “attachment that cannot be opened” (in absence of an OpenPGP-aware setup)
  - Do not (invasively) show such attachments to (ordinary) users
Use Case 2: pEp KeySync

- Users usually have multiple devices; received messages cannot be decrypted on all devices
  - Missing private key
  - Encrypted with private key from another device
- pEp KeySync: Synchronization of private Keys
  - In a secure, peer-to-peer manner
- (Such types of) Structured Emails sent between the devices
  - Form (join or leave) a Device Group
  - Share keys among members of a Device Group
UC2: e.g., Form Device Group (simplified)

"Requester"-Device

- Beacon (challenge TID)
- NegotiationRequest
- NegotiationOpen
- (Display Trustwords)

User

- Handshake (user comparison of Trustwords) successful
- Accept
- CommitAcceptRequester
- User presses Accept Button on Requester Device

"Offerer"-Device

- NegotiationRequest
- (Display Trustwords)
- Accept
- User presses Accept Button on Offerer Device
- CommitAcceptOfferer
- OwnKeysOfferer + keys

- Offerer is Grouped

Note: The roles are determined by the Transaction-ID (TID): lower TID gets Requester, higher TID gets Offerer; NegotiationRequest sent only if own TID is lower than challenge TID in the beacon.

Note: Beacon is sent by every device (e.g. on initialization) with challenge TID.

Note: Similar if User accepts first on "Offerer"-Device (only timing of messages different).

Note: OwnKeysRequestor is sent on processing CommitAcceptOfferer.

Green Arrows represent structured Email Messages.
Use Case 3: pEp KeyReset

- pEp KeyReset protocol is used to revoke/replace/rotate public keys; relevant also for Zero Trust environments, rotating often
- pEp KeyReset also plays an important role in group communications:
  - Managed groups (admin adding or removing members)
  - Unmanaged groups (with members leaving)
- This is done in a peer-to-peer manner, by sending (such types of) Structured Emails:
  - Distribute Information about new keys to be imported, alongside with revocation certificates
  - Such “technical messages” are intended to be processed automatically, not to be read by users
Summary and Requirements

Structured Emails or attachments might be received, not intended to be read by users (in full), but to be processed automatically.

This may confuse users. Mechanisms to improve usability:

1. Means to instruct receiving MUAs to not to render certain *attachments* to users.
2. Means to instruct receiving MUAs to not to display certain *emails* to users.
3. Means to instruct receiving MUAs to automatically process (or not process) an *attachment* (or MIME sub-tree) [optional].
Questions / Discussion

Privacy by Default.
Backup Slides
Example Msg. flow (simplified)

Privacy Status for B: Unencrypted
Auto-Generate key pair (if no key yet)
A sends message to B (public key attached), not encrypted
Privacy Status for A: Encrypted
B sends message to A (public key attached), encrypted
Privacy Status for B: Trusted

User A
Device A
Device B
User B

Write message
Read message
Display Trustwords
Confirm Trustwords

A sends message to B (public key attached), not encrypted
B sends message to A (public key attached), encrypted

User A and B compare their Trustwords over an alternative channel (e.g. phone line)
Handshake

Privacy Status for A: Encrypted
Privacy Status for A: Trusted
Auto-Generate key pair (if no key yet)

Write message
Read message
Display Trustwords
Confirm Trustwords

TOFU (Trust on first use)
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.

---

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--
This is a multi-part message in MIME format.

This message is part of pEp's concept to manage keys. You can safely ignore it. It will be deleted automatically.

The content types, character sets, and transfer encodings are as follows:

**Content-Type: application/pEp.sync; name="sync.pEp"**
Content-Disposition: attachment; filename="sync.pEp"
Content-Transfer-Encoding: base64

**Content-Type: application/pEp.sign; name="electronic_signature.asc"**
Content-Disposition: attachment; filename="electronic_signature.asc"
Content-Transfer-Encoding: base64

**Content-Type: application/pgp-keys; name="file://sender_key.asc"**
Content-Disposition: attachment; filename="file://sender_key.asc"
Content-Description: OpenPGP public key
Content-Transfer-Encoding: 7bit

-----BEGIN PGP PUBLIC KEY BLOCK-----
[...]
-----END PGP PUBLIC KEY BLOCK-----
From: alice@example.org
To: bob@example.org
Subject: pEp
X-pEp-Version: 2.1
X-pEp-autoconsume: yes
MIME-Version: 1.0
Content-Type: multipart/encrypted;
  boundary="boundary2";
  protocol="application/pgp-encrypted"

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

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

-----BEGIN PGP MESSAGE-----
[... {PGP-encrypted payload} ...]
-----END PGP MESSAGE-----

--boundary2
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: quoted-printable

Subject: pEp key management message - please ignore

This message is part of pEp's concept to manage keys.
You can safely ignore it. It will be deleted automatically.

--boundary2
Content-Type: application/pEp.distribution
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="ignore_this_attachment.pEp"

BAAEHnBlcC50ZXN0LmFsaWNlQHBlc1wcm9qZWN0Lm9yZzCVfHVes1hln8nwygFSgjou3+AdLhzgiuC+3u7cvmrnuSSyCaC2NL6ykDS3ECu3tzIyuTYwtzj/kaYmzzJQmbHcIKpDIB2akY+Q5y3E=

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

-----BEGIN PGP PUBLIC KEY BLOCK-----
[...]
-----END PGP PUBLIC KEY BLOCK-----

--boundary2--
Running Code

https://pep.software/

- p≡p for Outlook (release: add-on)
- p≡p for Android (release: app)
- p≡p for iOS (release: app)
- p≡p for Thunderbird (release: add-on)

Privacy by Default.
Join the discussion

- MEDUP (Missing Elements for Decentralized and Usable Privacy) side meeting at IETF-116
  - Room G-301 on Tue 2023-03-28 (today), 18:30-19:30 JST

- Mailing list discussion:
  - medup@ietf.org
  - To subscribe: https://www.ietf.org/mailman/listinfo/MEDUP

- Contact us directly:
  - hernani.marques@pep.foundation
  - bernie@ietf.hoeneisen.ch