Workgroup: lamps
Internet-Draft:
draft-ietf-lamps-e2e-mail-guidance-05
Published: 30 January 2023
Intended Status: Informational
Expires: 3 August 2023
Authors: D. K. Gillmor, Ed.
ACLU

Guidance on End-to-End E-mail Security

Abstract

End-to-end cryptographic protections for e-mail messages can provide useful security. However, the standards for providing cryptographic protection are extremely flexible. That flexibility can trap users and cause surprising failures. This document offers guidance for mail user agent implementers that need to compose or interpret e-mail messages with end-to-end cryptographic protection. It provides a useful set of vocabulary as well as suggestions to avoid common failures.

About This Document

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

The latest revision of this draft can be found at https://dkg.gitlab.io/e2e-mail-guidance/. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-lamps-e2e-mail-guidance/.

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

Source for this draft and an issue tracker can be found at https://gitlab.com/dkg/e2e-mail-guidance.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 August 2023.

Copyright Notice

Table of Contents

1. Introduction

E-mail end-to-end security using S/MIME ([RFC8551]) and PGP/MIME ([RFC3156]) cryptographic standards can provide integrity, authentication and confidentiality to MIME ([RFC4289]) e-mail messages.

However, there are many ways that a receiving mail user agent can misinterpret or accidentally break these security guarantees (e.g., [EFAIL]).

A mail user agent that interprets a message with end-to-end cryptographic protections needs to do so defensively, staying alert to different ways that these protections can be bypassed by mangling (either malicious or accidental) or a failed user experience.

A mail user agent that generates a message with end-to-end cryptographic protections should be aware of these defensive interpretation strategies, and should compose any new outbound message conservatively if they want the protections to remain intact.

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.

1.1. Terminology

For the purposes of this document, we define the following concepts:

* *MUA* is short for Mail User Agent; an e-mail client.

* *Protection* of message data refers to cryptographic encryption and/or signatures, providing confidentiality, authenticity, and/or integrity.

* *Cryptographic Layer*, *Cryptographic Envelope*, *Cryptographic Payload*, and *Errant Cryptographic Layer* are defined in Section 4

* A *well-formed* e-mail message with cryptographic protection has both a *Cryptographic Envelope* and a *Cryptographic Payload*.

* *Structural Headers* are documented in Section 1.1.1.

* *User-Facing Headers* are documented in Section 1.1.2.

* *Main Body Part* is the part (or parts) that are typically rendered to the user as the message itself (not "as an attachment"). See Section 7.1.

1.1.1. Structural Headers

A message header field named MIME-Version, or whose name begins with Content- is referred to in this document as a "structural" header. This is a less-ambiguous name for what [RFC2045] calls "MIME Header Fields".

These headers indicate something about the specific MIME part they are attached to, and cannot be transferred or copied to other parts without endangering the readability of the message.

This includes:

* MIME-Version

* Content-Type

* Content-Transfer-Encoding

* Content-Disposition

1.1.2. User-Facing Headers

Of all the headers that an e-mail message may contain, only a handful are typically presented directly to the user. The user-facing headers are:

* Subject

* From

* To

* Cc

* Date

*Reply-To

*Followup-To

The above is a complete list. No other headers are considered "user-facing".

Other headers may affect the visible rendering of the message (e.g., References and In-Reply-To may affect the placement of a message in a threaded discussion), but they are not directly displayed to the user and so are not considered "user-facing".

2. Usability

Any MUA that enables its user to transition from unprotected messages to messages with end-to-end cryptographic protection needs to consider how the user understands this transition. That said, the primary goal of the user of an MUA is communication -- so interface elements that get in the way of communication should be avoided where possible.

Furthermore, it is likely is that the user will continue to encounter unprotected messages, and may need to send unprotected messages (for example, if a given recipient cannot handle cryptographic protections). This means that the MUA needs to provide the user with some guidance, so that they understand what protections any given message or conversation has. But the user should not be overwhelmed with choices or presented with unactionable information.

2.1. Simplicity

The end user (the operator of the MUA) is unlikely to understand complex end-to-end cryptographic protections on any e-mail message, so keep it simple.

For clarity to the user, any cryptographic protections should apply to the message as a whole, not just to some subparts.

This is true for message composition: the standard message composition user interface of an MUA should offer minimal controls which indicate which types of protection to apply to the new message as a whole.

This is also true for message interpretation: the standard message rendering user interface of an MUA should offer a minimal, clear indicator about the end-to-end cryptographic status of the message as a whole.

See Section 3 for more detail about mental models and cryptographic status.

2.2. E-mail Users Want a Familiar Experience

A person communicating over the Internet today often has many options for reaching their desired correspondent, including web-based bulletin boards, contact forms, and instant messaging services.

E-mail offers a few distinctions from these other systems, most notably features like:

* Ubiquity: Most correspondents will have an e-mail address, while not everyone is present on every alternate messaging service,

* Federation: interaction between users on distinct domains who have not agreed on a common communications provider is still possible, and

* User Control: the user can interact with the e-mail system using a MUA of their choosing, including automation and other control over their preferred and/or customized workflow.

Other systems (like some popular instant messaging applications, such as WhatsApp and Signal Private Messenger) offer built-in end-to-end cryptographic protections by default, which are simpler for the user to understand. ("All the messages I see on Signal are confidential and integrity-protected" is a clean user story)

A user of e-mail is likely using e-mail instead of other systems because of the distinctions outlined above. When adding end-to-end cryptographic protection to an e-mail endpoint, care should be taken not to negate any of the distinct features of e-mail as a whole. If these features are violated to provide end-to-end crypto, the user may just as well choose one of the other systems that don't have the drawbacks that e-mail has. Implmenters should try to provide end-to-end protections that retain the familiar experience of e-mail itself.

Furthermore, an e-mail user is likely to regularly interact with other e-mail correspondents who *cannot* handle or produce end-to-end cryptographic protections. Care should be taken that enabling cryptography in a MUA does not inadvertently limit the ability of the user to interact with legacy correspondents.

2.3. Warning About Failure vs. Announcing Success

Moving the web from http to https offers useful historical similarities to adding end-to-end encryption to e-mail.

In particular, the indicators of what is "secure" vs. "insecure" for web browsers have changed over time. For example, years ago the default experience was http, and https sites were flagged with "secure" indicators like a lock icon. In 2018, some browsers reversed that process by downplaying https, and instead visibly marking http as "not secure" (see [chrome-indicators]).

By analogy, when the user of a MUA first enables end-to-end cryptographic protection, it's likely that they will want to see which messages *have* protection. But a user whose e-mail communications are entirely end-to-end protected might instead want to know which messages do *not* have the expected protections.

Note also that some messages are expected to be confidential, but other messages are expected to be public -- the types of protection (see Section 3) that apply to each particular message will be different. And the types of protection that are *expected* to be present in any context might differ (for example, by sender, by thread, or by date).

It is out of scope for this document to define expectations about protections for any given message, but an implementer who cares about usable experience should be deliberate and judicious about the expectations their interface assumes that the user has in a given context.

3. Types of Protection

A given message might be:

* signed,

* encrypted,

* both signed and encrypted, or

* none of the above.

Given that many e-mail messages offer no cryptographic protections, the user needs to be able to detect which protections are present for any given message.

3.1. Simplified Mental Model

To the extent that an e-mail message actually does have end-to-end cryptographic protections, those protections need to be visible and comprehensible to the end user. If the user is unaware of the protections, then they do not extend all the way to the "end".

However, most users do not have (or want to have) a sophisticated mental model of what kinds of protections can be associated with a given message. Even the four states above approach the limits of complexity for an interface for normal users.

While Section 5.3 recommends avoiding deliberate creation of encrypted-only messages, some messages may end up in the encrypted-only state due to signature failure or certificate revocation.

A simple model for the user could be that a message is in one of three normal states:

* Unprotected

* Verified (has a valid signature from the apparent sender of the message)

* Confidential (meaning, encrypted, with a valid signature from the apparent sender of the message)

And one error state:

* Encrypted But Unverified (meaning, encrypted without a valid signature from the apparent sender of the message)

Note that this last state is not "Confidential" (a secret shared exclusively between the participants in the communication) because the recipient does not know for sure who sent it.

In an ecosystem where encrypted-only messages are never deliberately sent (see Section 5.3), representing an Encrypted But Unverified message as a type of user-visible error is not unreasonable.

Alternately, a MUA may prefer to represent the state of a Encrypted but Unverified message to the user as though it was Unprotected, since no verification is possible. However the MUA represents the message to the user, though, it **MUST NOT** leak cleartext of an encrypted message (even an Encrypted but Unverified message) in subsequent replies (see Section 5.4) or similar replications of the message.

Note that a cleartext message with an invalid signature **SHOULD NOT** be represented to the user as anything other than Unprotected (see Section 6.4).

In a messy legacy ecosystem, a MUA may prefer instead to represent "Signed" and "Encrypted" as orthogonal states of any given message, at the cost of an increase in the complexity of the user's mental model.

3.2. One Cryptographic Status Per Message

Some MUAs may attempt to generate multiple copies of a given e-mail message, with different copies offering different types of protection (for example, opportunistically encrypting on a per-recipient basis). A message resulting from this approach will have a cryptographic state that few users will understand. Even if the sender understands the different statuses of the different copies, the recipients of the messages may not understand (each recipient might not even know about the other copies). See for example the discussion in Section 9.3 for how this can go wrong.

For comprehensibility, a MUA **SHOULD NOT** create multiple copies of a given message that differ in the types of end-to-end cryptographic protections afforded.

For opportunistic cryptographic protections that are not surfaced to the user (that is, that are not end-to-end), other mechanisms like transport encryption ([RFC3207]) or domain-based signing ([RFC6376]) may be preferable. These opportunistic protections are orthogonal to the end-to-end protections described in this document.

To the extent that opportunistic protections are made visible to the user for a given copy of a message, a reasonable MUA will distinguish that status from the message's end-to-end cryptographic status. But the potential confusion caused by rendering this complex, hybrid state may not be worth the value of additional knowledge gained by the user. The benefits of opportunistic protections accrue (or don't) even without visibility to the user.

The user needs a single clear, simple, and correct indication about the end-to-end cryptographic status of any given message.

4. Cryptographic MIME Message Structure

Implementations use the structure of an e-mail message to establish (when sending) and understand (when receiving) the cryptographic status of the message. This section establishes some conventions about how to think about message structure.

4.1. Cryptographic Layers

"Cryptographic Layer" refers to a MIME substructure that supplies some cryptographic protections to an internal MIME subtree. The internal subtree is known as the "protected part" though of course it may itself be a multipart object.

In the diagrams below, "↧" (DOWNWARDS ARROW FROM BAR, U+21A7) indicates "decrypts to", and "⇩" (DOWNWARDS WHITE ARROW, U+21E9) indicates "unwraps to".

4.1.1. S/MIME Cryptographic Layers

For S/MIME [RFC8551], there are four forms of Cryptographic Layers: multipart/signed, PKCS#7 signed-data, PKCS7 enveloped-data, PKCS7 authEnveloped-data.

4.1.1.1. S/MIME Multipart Signed Cryptographic Layer

```
└┬─multipart/signed; protocol="application/pkcs7-signature"
 ├── [protected part]
 └── application/pkcs7-signature
```

This MIME layer offers authentication and integrity.

4.1.1.2. S/MIME PKCS7 signed-data Cryptographic Layer

```
└── application/pkcs7-mime; smime-type="signed-data"
 ⇩ (unwraps to)
 └── [protected part]
```

This MIME layer offers authentication and integrity.

4.1.1.3. S/MIME PKCS7 enveloped-data Cryptographic Layer

```
└── application/pkcs7-mime; smime-type="enveloped-data"
 ↧ (decrypts to)
 └── [protected part]
```

This MIME layer offers confidentiality.

4.1.1.4. S/MIME PKCS7 authEnveloped-data Cryptographic Layer

```
└── application/pkcs7-mime; smime-type="authEnveloped-data"
 ↧ (decrypts to)
 └── [protected part]
```

This MIME layer offers confidentiality and integrity.

Note that enveloped-data (Section 4.1.1.3) and authEnveloped-data (Section 4.1.1.4) have identical message structure and very similar semantics. The only difference between the two is ciphertext malleability.

The examples in this document only include enveloped-data, but the implications for that layer apply to authEnveloped-data as well.

4.1.1.5. PKCS7 Compression is NOT a Cryptographic Layer

The Cryptographic Message Syntax (CMS) provides a MIME compression layer (smime-type="compressed-data"), as defined in [RFC3274]. While

the compression layer is technically a part of CMS, it is not considered a Cryptographic Layer for the purposes of this document.

4.1.2. PGP/MIME Cryptographic Layers

For PGP/MIME [RFC3156] there are two forms of Cryptographic Layers, signing and encryption.

4.1.2.1. PGP/MIME Signing Cryptographic Layer (multipart/signed)

```
└┬─multipart/signed; protocol="application/pgp-signature"
 ├── [protected part]
 └── application/pgp-signature
```

This MIME layer offers authenticity and integrity.

4.1.2.2. PGP/MIME Encryption Cryptographic Layer (multipart/encrypted)

```
└┬─multipart/encrypted
 ├── application/pgp-encrypted
 └── application/octet-stream
  ↧ (decrypts to)
  └── [protected part]
```

This MIME layer can offer any of:

- confidentiality (via a Symmetrically Encrypted Data Packet, see Section 5.7 of [RFC4880]; a MUA **MUST NOT** generate this form due to ciphertext malleability)

- confidentiality and integrity (via a Symmetrically Encrypted Integrity Protected Data Packet (SEIPD), see Section 5.13 of [RFC4880]), or

- confidentiality, integrity, and authenticity all together (by including an OpenPGP Signature Packet within the SEIPD).

4.2. Cryptographic Envelope

The Cryptographic Envelope is the largest contiguous set of Cryptographic Layers of an e-mail message starting with the outermost MIME type (that is, with the Content-Type of the message itself).

If the Content-Type of the message itself is not a Cryptographic Layer, then the message has no cryptographic envelope.

"Contiguous" in the definition above indicates that if a Cryptographic Layer is the protected part of another Cryptographic Layer, the layers together comprise a single Cryptographic Envelope.

Note that if a non-Cryptographic Layer intervenes, all Cryptographic Layers within the non-Cryptographic Layer *are not* part of the Cryptographic Envelope. They are Errant Cryptographic Layers (see Section 4.5).

Note also that the ordering of the Cryptographic Layers implies different cryptographic properties. A signed-then-encrypted message is different than an encrypted-then-signed message. See Section 5.2.

4.3. Cryptographic Payload

The Cryptographic Payload of a message is the first non-Cryptographic Layer -- the "protected part" -- within the Cryptographic Envelope.

4.4. Types of Cryptographic Envelope

4.4.1. Simple Cryptographic Envelopes

As described above, if the "protected part" identified in the section above is not itself a Cryptographic Layer, that part *is* the Cryptographic Payload.

If the application wants to generate a message that is both encrypted and signed, it **MAY** use the simple MIME structure from Section 4.1.2.2 by ensuring that the [RFC4880] Encrypted Message within the application/octet-stream part contains an [RFC4880] Signed Message (the final option described in Section 4.1.2.2.

4.4.2. Multilayer Cryptographic Envelopes

It is possible to construct a Cryptographic Envelope consisting of multiple layers with either S/MIME or PGP/MIME , for example using the following structure:

```
A └─ application/pkcs7-mime; smime-type="enveloped-data"
B  ↧ (decrypts to)
C  └─ application/pkcs7-mime; smime-type="signed-data"
D   ⇩ (unwraps to)
E   └─ [protected part]
```

When handling such a message, the properties of the Cryptographic Envelope are derived from the series A, C.

As noted in Section 4.4.1, PGP/MIME applications also have a simpler MIME construction available with the same cryptographic properties.

4.5. Errant Crytptographic Layers

Due to confusion, malice, or well-intentioned tampering, a message may contain a Cryptographic Layer that is not part of the Cryptographic Envelope. Such a layer is an Errant Cryptographic Layer.

An Errant Cryptographic Layer **SHOULD NOT** contribute to the message's overall cryptographic state.

Guidance for dealing with Errant Cryptographic Layers can be found in Section 6.2.

4.5.1. Mailing List Wrapping

Some mailing list software will re-wrap a well-formed signed message before re-sending to add a footer, resulting in the following structure seen by recipients of the e-mail:

```
H └┬ multipart/mixed
I  ├┬ multipart/signed
J  │├── text/plain
K  │└── application/pgp-signature
L  └── text/plain
```

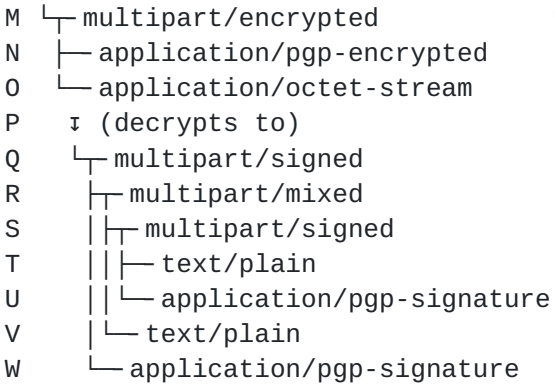
In this message, L is the footer added by the mailing list. I is now an Errant Cryptographic Layer.

Note that this message has no Cryptographic Envelope at all.

It is **NOT RECOMMENDED** to produce e-mail messages with this structure, because the data in part L may appear to the user as though it were part of J, though they have different cryptographic properties. In particular, if the user believes that the message is signed, but cannot distinguish L from J then the author of L can effectively tamper with content of the signed message, breaking the user's expectation of integrity and authenticity.

4.5.2. A Baroque Example

Consider a message with the following overcomplicated structure:



The 3 Cryptographic Layers in such a message are rooted in parts M, Q, and S. But the Cryptographic Envelope of the message consists only of the properties derived from the series M, Q. The Cryptographic Payload of the message is part R. Part S is an Errant Cryptographic Layer.

Note that this message has both a Cryptographic Envelope *and* an Errant Cryptographic Layer.

It is **NOT RECOMMENDED** to generate messages with such complicated structures. Even if a receiving MUA can parse this structure properly, it is nearly impossible to render in a way that the user can reason about the cryptographic properties of part T compared to part V.

5. Message Composition

This section describes the ideal composition of an e-mail message with end-to-end cryptographic protection. A message composed with this form is most likely to achieve its end-to-end security goals.

5.1. Message Composition Algorithm

This section roughly describes the steps that a MUA should use to compose a cryptographically-protected message that has a proper cryptographic envelope and payload.

The message composition algorithm takes three parameters:

* origbody: the traditional unprotected message body as a well-formed MIME tree (possibly just a single MIME leaf part). As a well-formed MIME tree, origbody already has structural headers present (see Section 1.1.1).

* origheaders: the intended non-structural headers for the message, represented here as a list of (h,v) pairs, where h is a header field name and v is the associated value.

*crypto: The series of cryptographic protections to apply (for example, "sign with the secret key corresponding to X.509 certificate X, then encrypt to X.509 certificates X and Y"). This is a routine that accepts a MIME tree as input (the Cryptographic Payload), wraps the input in the appropriate Cryptographic Envelope, and returns the resultant MIME tree as output.

The algorithm returns a MIME object that is ready to be injected into the mail system:

*Apply crypto to origbody, yielding MIME tree output

*For each header name and value (h,v) in origheaders:

-Add header h of output with value v

*Return output

5.2. Encryption Outside, Signature Inside

Users expect any message that is both signed and encrypted to be signed *inside* the encryption, and not the other way around.

Putting the signature inside the encryption has two advantages:

*The details of the signature remain confidential, visible only to the parties capable of decryption.

*Any mail transport agent that modifies the message is unlikely to be able to accidentally break the signature.

A MUA **SHOULD NOT** generate an encrypted and signed message where the only signature is outside the encryption.

5.3. Avoid Offering Encrypted-only Messages

When generating an e-mail, the user has options about what forms of end-to-end cryptographic protections to apply to it.

In some cases, offering any end-to-end cryptographic protection is harmful: it may confuse the recipient and offer no benefit.

In other cases, signing a message is useful (authenticity and integrity are desirable) but encryption is either impossible (for example, if the sender does not know how to encrypt to all recipients) or meaningless (for example, an e-mail message to a mailing list that is intended to be be published to a public archive).

In other cases, full end-to-end confidentiality, authenticity, and integrity are desirable.

It is unclear what the use case is for an e-mail message with end-to-end confidentiality but without authenticity or integrity.

A reasonable MUA will keep its message composition interface simple, so when presenting the user with a choice of cryptographic protection, it **SHOULD** offer no more than three choices:

- no end-to-end cryptographic protection
- Verified (signed only)
- Confidential (signed and encrypted)

Note that these choices correspond to the simplified mental model in Section 3.1.

5.4. Composing a Reply Message

When replying to a message, most MUAs compose an initial draft of the reply that contains quoted text from the original message. A responsible MUA will take precautions to avoid leaking the cleartext of an encrypted message in such a reply.

If the original message was end-to-end encrypted, the replying MUA **MUST** either:

- compose the reply with end-to-end encryption, or
- avoid including quoted text from the original message.

In general, MUAs **SHOULD** prefer the first option: to compose an encrypted reply. This is what users expect.

However, in some circumstances, the replying MUA cannot compose an encrypted reply. For example, the MUA might not have a valid, unexpired, encryption-capable certificate for all recipients. This can also happen during composition when a user adds a new recipient into the reply, or manually toggles the cryptographic protections to remove encryption.

In this circumstance, the composing MUA **SHOULD** strip the quoted text from the original message.

Note additional nuance about replies to malformed messages that contain encryption in Section 6.2.2.1.

6. Message Interpretation

Despite the best efforts of well-intentioned senders to create e-mail messages with well-formed end-to-end cryptographic protection, receiving MUAs will inevitably encounter some messages with malformed end-to-end cryptographic protection.

This section offers guidance on dealing with both well-formed and malformed messages containing Cryptographic Layers.

6.1. Rendering Well-formed Messages

A message is well-formed when it has a Cryptographic Envelope, a Cryptographic Payload, and no Errant Cryptographic Layers. Rendering a well-formed message is straightforward.

The receiving MUA should evaluate and summarize the cryptographic properties of the Cryptographic Envelope, and display that status to the user in a secure, strictly-controlled part of the UI. In particular, the part of the UI used to render the cryptographic summary of the message **MUST NOT** be spoofable, modifiable, or otherwise controllable by the received message itself.

Aside from this cryptographic summary, the message itself should be rendered as though the Cryptographic Payload is the body of the message. The Cryptographic Layers themselves **SHOULD** not be rendered otherwise.

6.2. Errant Cryptographic Layers

If an incoming message has any Errant Cryptographic Layers, the interpreting MUA **SHOULD** ignore those layers when rendering the cryptographic summary of the message to the user.

6.2.1. Errant Signing Layer

When rendering a message with an Errant Cryptographic Layer that provides authenticity and integrity (via signatures), the message should be rendered by replacing the Cryptographic layer with the part it encloses.

For example, a message with this structure:

```
A └┬ multipart/mixed
B  ├ text/plain
C  ├┬ multipart/signed
D  │├ image/jpeg
E  │└ application/pgp-signature
F  └ text/plain
```

Should be rendered identically to this:

```
A └┬ multipart/mixed
B  ├─ text/plain
D  ├─ image/jpeg
F  └─ text/plain
```

In such a situation, an MUA **SHOULD NOT** indicate in the cryptographic summary that the message is signed.

6.2.1.1. Exception: Mailing List Footers

The use case described in Section 4.5.1 is common enough in some contexts, that a MUA **MAY** decide to handle it as a special exception.

If the MUA determines that the message comes from a mailing list (for example, it has a List-ID header), and it has a structure that appends a footer to a signing-only Cryptographic Layer with a valid signature, such as:

```
H └┬ multipart/mixed
I  ├┬ multipart/signed
J  │├─ [protected part, may be arbitrary MIME subtree]
K  │└─ application/{pgp,pkcs7}-signature
L  └─ [footer, typically text/plain]
```

or:

```
H └┬ multipart/mixed
I  ├─ application/pkcs7-mime; smime-type="signed-data"
   │⇩ (unwraps to)
J  │└─ [protected part, may be an arbitrary MIME subtree]
L  └─ [footer, typically text/plain]
```

Then, the MUA **MAY** indicate to the user that this is a signed message that has been wrapped by the mailing list.

In this case, the MUA **MUST** distinguish the footer (part L) from the protected part (part J) when rendering any information about the signature.

One way to do this is to offer the user two different views of the message: the "mailing list" view, which hides any cryptographic summary but shows the footer:

```
Cryptographic Protections: none
H └┬ multipart/mixed
J  ├─ [protected part, may be arbitrary MIME subtree]
L  └─ [footer, typically text/plain]
```

or the "sender's view", which shows the cryptographic summary but hides the footer:

```
Cryptographic Protections: signed [details from part I]
J └── [protected part, may be arbitrary MIME subtree]
```

6.2.2. Errant Encryption Layer

An MUA may encounter a message with an Errant Cryptographic Layer that offers confidentiality (encryption), and the MUA is capable of decrypting it.

The user wants to be able to see the contents of any message that they receive, so an MUA in this situation **SHOULD** decrypt the part.

In this case, though, the MUA **MUST NOT** indicate in the message's cryptographic summary that the message itself was encrypted. Such an indication could be taken to mean that other (non-encrypted) parts of the message arrived with cryptographic confidentiality.

Furthermore, when decrypting an Errant Cryptographic Layer, the MUA **MUST** treat the decrypted cleartext as a distinct MIME subtree, and not attempt to merge or splice it together with any other part of the message. This offers protection against the direct exfiltration (also known as EFAIL-DE) attacks described in [EFAIL] and so-called multipart/oracle attacks described in [ORACLE].

6.2.2.1. Replying to a Message with an Errant Encryption Layer

Note that there is an asymmetry here between rendering and replying to a message with an Errant Encryption Layer.

When rendering, the MUA does not indicate that the message was encrypted, even if some subpart of it was decrypted for rendering.

When composing a reply to a message that has any encryption layer, even an errant one, the reply message **SHOULD** be marked for encryption, as noted in {#composing-reply}.

When composing a reply to a message with an errant cryptographic layer, the MUA **MUST NOT** decrypt any errant cryptographic layers when generating quoted or attributed text. This will typically mean either leaving the ciphertext itself in the generated reply message, or simply no generating any quoted or attributed text at all. This offers protection against the reply-based attacks described in [EFAIL].

In all circumstances, if the reply message cannot be encrypted (or if the user elects to not encrypt the reply), the composed reply **MUST NOT** include any material from the decrypted subpart.

6.2.3. Avoiding Non-MIME Cryptographic Mechanisms

In some cases, there may be a cryptographic signature or encryption that does not coincide with a MIME boundary. For example so-called "PGP Inline" messages typically contain base64-encoded ("ASCII-armored", see Section 6 of [RFC4880]) ciphertext, or within the content of a MIME part.

6.2.3.1. Do Not Validate Non-MIME Signatures

When encountering cryptographic signatures in these positions, a MUA **MUST NOT** attempt to validate any signature. It is challenging to communicate to the user exactly which part of such a message is covered by the signature, so it is better to leave the message marked as unsigned.

6.2.3.2. Skip or Isolate Non-MIME Decryption When Rendering

When encountering what appears to be encrypted data not at a MIME boundary, the MUA **MAY** decline to decrypt the data at all.

During message rendering, if the MUA attempts decryption of such a non-MIME encrypted section of an e-mail, it **MUST** synthesize a separate MIME part to contain only the decrypted data, and not attempt to merge or splice that part together with any other part of the message. Keeping such a section distinct and isolated from any other part of the message offers protection against the direct exfiltration attacks (also known as EFAIL-DE) described in [EFAIL].

6.2.3.3. Do Not Decrypt Non-MIME Decryption when Replying

When composing a reply to a message with such a non-MIME encrypted section, the MUA **MUST NOT** decrypt the any non-MIME encrypted section when generating quoted or attributed text, similar to the guidance in Section 6.2.2.1.

This offers protection against the reply-based attacks described in [EFAIL].

6.3. Forwarded Messages with Cryptographic Protection

An incoming e-mail message may include an attached forwarded message, typically as a MIME subpart with Content-Type: message/rfc822 ([RFC5322]) or Content-Type: message/global ([RFC5355]).

Regardless of the cryptographic protections and structure of the incoming message, the internal forwarded message may have its own Cryptographic Envelope.

The Cryptographic Layers that are part of the Cryptographic Envelope of the forwarded message are not Errant Cryptographic Layers of the surrounding message -- they are simply layers that apply to the forwarded message itself.

The rendering MUA **MUST NOT** conflate the cryptographic protections of the forwarded message with the cryptographic protections of the incoming message.

The rendering MUA **MAY** render a cryptographic summary of the protections afforded to the forwarded message by its own Cryptographic Envelope, as long as that rendering is unambiguously tied to the forwarded message itself, and cannot be spoofed either by the enclosing message or by the forwarded message.

6.4. Signature failures

A cryptographic signature may fail in multiple ways. A receiving MUA that discovers a failed signature should treat the message as though the signature did not exist. This is similar to the standard guidance for about failed DKIM signatures (see Section 6.1 of [RFC6376]).

A MUA **SHOULD NOT** render a message with a failed signature as more dangerous or more dubious than a comparable message without any signature at all.

A MUA that encounters an encrypted-and-signed message where the signature is invalid **SHOULD** treat the message the same way that it would treat a message that is encryption-only.

Some different ways that a signature may be invalid on a given message:

* the signature is not cryptographically valid (the math fails).

* the signature relies on suspect cryptographic primitives (e.g. over a legacy digest algorithm, or was made by a weak key, e.g., 1024-bit RSA)

* the signature is made by a certificate which the receiving MUA does not have access to.

* the certificate that made the signature was revoked.

* the certificate that made the signature was expired at the time that the signature was made.

* the certificate that made the signature does not correspond to the author of the message. (for X.509, there is no subjectAltName

of type RFC822Name whose value matches an e-mail address found in From: or Sender:)

* the certificate that made the signature was not issued by an authority that the MUA user is willing to rely on for certifying the sender's e-mail address, and the user has no other reasonable indication that the certificate belongs to the sender's e-mail address.

* the signature indicates that it was made at a time much before or much after from the date of the message itself.

A valid signature must pass all these tests, but of course invalid signatures may be invalid in more than one of the ways listed above.

7. Reasoning about Message Parts

When generating or rendering messages, it is useful to know what parts of the message are likely to be displayed, and how. This section introduces some common terms that can be applied to parts within the Cryptographic Payload.

7.1. Main Body Part

When an e-mail message is composed or rendered to the user there is typically one main view that presents a (mostly textual) part of the message.

While the message itself may be constructed of several distinct MIME parts in a tree, the part that is rendered to the user is the "Main Body Part".

When rendering a message, one of the primary jobs of the receiving MUA is identifying which part (or parts) is the Main Body Part. Typically, this is found by traversing the MIME tree of the message looking for a leaf node that has a primary content type of text (e.g. text/plain or text/html) and is not Content-Disposition: attachment.

MIME tree traversal follows the first child of every multipart node, with the exception of multipart/alternative. When traversing a multipart/alternative node, all children should be scanned, with preference given to the last child node with a MIME type that the MUA is capable of rendering directly.

A MUA **MAY** offer the user a mechanism to prefer a particular MIME type within multipart/alternative instead of the last renderable child. For example, a user may explicitly prefer a text/plain alternative part over text/html.

Note that due to uncertainty about the capabilities and configuration of the receiving MUA, the composing MUA **SHOULD** consider that multiple parts might be rendered as the Main Body Part when the message is ultimately viewed.

When composing a message, an originating MUA operating on behalf of an active user can identify which part (or parts) are the "main" parts: these are the parts the MUA generates from the user's editor. Tooling that automatically generates e-mail messages should also have a reasonable estimate of which part (or parts) are the "main" parts, as they can be programmatically identified by the message author.

For a filtering program that attempts to transform an outbound message without any special knowledge about which parts are Main Body Parts, it can identify the likely parts by following the same routine as a receiving MUA.

7.2. Attachments

A message may contain one or more separated MIME parts that are intended for download or extraction. Such a part is commonly called an "attachment", and is commonly identified by having Content-Disposition: attachment, and is a subpart of a multipart/mixed or multipart/related container.

An MUA **MAY** identify a subpart as an attachment, or permit extraction of a subpart even when the subpart does not have Content-Disposition: attachment.

For a message with end-to-end cryptographic protection, any attachment ***MUST*** be included within the Cryptographic Payload. If an attachment is found outside the Cryptographic Payload, then the message is not well-formed (see Section 6.1).

Some MUAs have tried to compose messages where each attachment is placed in its own cryptographic envelope. Such a message is problematic for several reasons:

* The attachments can be stripped, replaced, or reordered without breaking any cryptographic integrity mechanism.

* The resulting message may have a mix of cryptographic statuses (e.g. if a signature on one part fails but another succeeds, or if one part is encrypted and another is not). This mix of statuses is difficult to represent to the user in a comprehensible way.

7.3. MIME Part Examples

Consider a common message with the folloiwing MIME structure:

```
M └── application/pkcs7-mime
  ↧ (decrypts to)
N  └── application/pkcs7-mime
   ⇩ (unwraps to)
O   └┬─ multipart/mixed
P    ├┬─ multipart/alternative
Q    │├── text/plain
R    │└── text/html
S    └── image/png
```

Parts M and N comprise the Cryptographic Envelope.

Parts Q and R are both Main Body Parts.

If part S is Content-Disposition: attachment, then it is an attachment. If part S has no Content-Disposition header, it is potentially ambiguous whether it is an attachment or not.

Consider also this alternate structure:

```
M └── application/pkcs7-mime
  ↧ (decrypts to)
N  └── application/pkcs7-mime
   ⇩ (unwraps to)
O   └┬─ multipart/alternative
P    ├── text/plain
Q    └┬─ multipart/related
R     ├── text/html
S     └── image/png
```

In this case, parts M and N are still the Cryptographic Envelope.

Parts P and R (the first two leaf nodes within each subtree of part O) are the Main Body Parts.

Part S is more likely not to be an attachment, as the subtree layout suggests that it is only relevant for the HTML version of the message. For example, it might be rendered as an image within the HTML alternative.

8. Certificate Management

A cryptographically-capable MUA typically maintains knowledge about certificates for the user's own account(s), as well as certificates for the peers that it communicates with.

8.1. Peer Certificates

Most certificates that a cryptographically-capable MUA will use will be certificates belonging to the parties that the user communicates with through the MUA. This section discusses how to manage the certificates that belong to such a peer.

The MUA will need to be able to discover X.509 certificates for each peer, cache them, and select among them when composing an encrypted message.

8.1.1. Cert Discovery from Incoming Messages

TODO: incoming PKCS#7 messages tend to have a bundle of certificates in them. How should these certs be handled?

TODO: point to Autocrypt certificate discovery mechanism

TODO: point to OpenPGP embedded certificate subpacket proposal

TODO: compare mechanisms, explain where each case is useful.

8.1.2. Certificate Directories

Some MUAs may have the capability to look up peer certificates in a directory.

TODO: more information here about X.509 directories -- LDAP?

TODO: mention WKD for OpenPGP certificates?

TODO: mention SMIMEA and OPENPGPKEY DNS RRs

8.1.3. Peer Certificate Selection

When composing an encrypted message, the MUA needs to select a certificate for each recipient that is capable of encryption.

To select such a certificate for a given destination e-mail address, the MUA should look through all of its known certificates and verify that *all* of the conditions below are met:

* The certificate must be valid, not expired or revoked.

* It must have a subjectAltName of type rFC822Name whose contents exactly match the destination address.

* The algorithm OID in the certificate's SPKI is known to the MUA and capable of encryption. Examples include (TODO: need OIDs)

  - RSA, with keyUsage present and the "key encipherment" bit set

  - EC Public Key, with keyUsage present and the "key agreement" bit set

  - EC DH, with keyUsage present and the "key agreement" bit set

* If extendedKeyUsage is present, it contains at least one of the following OIDs: e-mail protection, anyExtendedKeyUsage.

TODO: If OID is EC Public Key and keyUsage is absent, what should happen?

TODO: what if multiple certificates meet all of these criteria for a given recipient?

8.1.4. Checking for Revocation

TODO: discuss how/when to check for peer certificate revocation

TODO: privacy concerns: what information leaks to whom when checking peer cert revocations?

8.2. Local Certificates

The MUA also needs to know about one or more certificates associated with the user's e-mail account. It is typically expected to have access to the secret key material associated with the public keys in those certificates.

8.2.1. Getting a Certificate for the User

TODO: mention ACME SMIME?

TODO: mention automatic self-signed certs e.g. OpenPGP?

TODO: **SHOULD** generate secret key material locally, and **MUST NOT** accept secret key material from an untrusted third party as the basis for the user's certificate.

8.2.2. Local Certificate Maintenance

The MUA should warn the user when/if:

* The user's own certificate set does not include a valid, unexpired encryption-capable X.509 certificate, and a valid, unexpired signature-capable X.509 certificate.

* Any of the user's own certificates is due to expire soon (TODO: what is "soon"?)

* Any of the user's own certificates does not match the e-mail address associated with the user's account.

* Any of the user's own certificates does not have a keyUsage section

* Any of the user's own certificates does not contain an extendedKeyUsage extension

TODO: how does the MUA do better than warning in the cases above? What can the MUA actually *do* here to fix problems before they happen?

TODO: discuss how/when to check for own certificate revocation, and what to do if it (or any intermediate certificate authority) is found to be revoked.

8.2.3. Shipping Certificates in Outbound Messages

TODO: What certificates should the MUA include in an outbound message so that peers can discover them?

* local signing certificate so that signature can be validated

* local encryption-capable certificate(s) so that incoming messages can be encrypted.

* On an encrypted message to multiple recipients, the encryption-capable peer certs of the other recipients (to enable "reply all")?

* intermediate certificates to chain all of the above to some set of root authorities?

8.3. Certificate Authorities

TODO: how should the MUA select root certificate authorities?

TODO: should the MUA cache intermediate CAs?

TODO: should the MUA share such a cache with other PKI clients (e.g., web browsers)? Are there distinctions between a CA for S/MIME and for the web?

9. Common Pitfalls and Guidelines

This section highlights a few "pitfalls" and guidelines based on these discussions and lessons learned.

FIXME: some possible additional commentary on:

- *indexing and search of encrypted messages
- *managing access to cryptographic secret keys that require user interaction
- *secure deletion
- *storage of composed/sent messages
- *cached signature validation
- *aggregated cryptographic status of threads/conversations ?
- *Draft messages
- *copies to the Sent folder

9.1. Reading Sent Messages

When composing a message, a typical MUA will store a copy of the message sent in sender's Sent mail folder so that the sender can read it later. If the message is an encrypted message, storing it encrypted requires some forethought to ensure that the sender can read it in the future.

It is a common and simple practice to encrypt the message not only to the recipients of the message, but also to the sender. One advantage of doing this is that the message that is sent on the wire can be identical to the message stored in the sender's Sent mail folder. This allows the sender to review and re-read the message even though it was encrypted.

There are at least three other approaches that are possible to ensure future readability by the sender of the message, but with different tradeoffs:

- *Encrypt two versions of the message: one to the recipients (this version is sent on the wire), and one to the sender only (this version is stored in the sender's Sent folder). This approach means that the message stored in the Sent folder is not byte-for-byte identical to the message sent to the recipients. In the event that message delivery has a transient failure, the MUA

cannot simply re-submit the stored message into the SMTP system and expect it to be readable by the recipient.

* Store a cleartext version of the message in the Sent folder. This presents a risk of information leakage: anyone with access to the Sent folder can read the contents of the message. Furthermore, any attempt to re-send the message needs to also re-apply the cryptographic transformation before sending, or else the message contents will leak upon re-send.

* A final option is that the MUA can store a copy of the message's encryption session key. Standard e-mail encryption mechanisms (e.g. S/MIME and PGP/MIME) are hybrid mechanisms: the asymmetric encryption steps simply encrypt a symmetric "session key", which is used to encrypt the message itself. If the MUA stores the session key itself, it can use the session key to decrypt the Sent message without needing the Sent message to be decryptable by the user's own asymmetric key. An MUA doing this must take care to store (and backup) its stash of session keys, because if it loses them it will not be able to read the sent messages; and if someone else gains access to them, they can decrypt the sent messages. This has the additional consequence that any other MUA accessing the same Sent folder cannot decrypt the message unless it also has access to the stashed session key.

9.2. Composing an Encrypted Message with Bcc

When composing an encrypted message containing at least one recipient address in the Bcc header field, there is a risk that the encrypted message itself could leak information about the actual recipients, even if the Bcc header field does not mention the recipient. For example, if the message clearly indicates which certificates it is encrypted to, the set of certificates can identify the recipients even if they are not named in the message headers.

Because of these complexities, there are several interacting factors that need to be taken into account when composing an encrypted message with Bcc'ed recipients.

* Section 3.6.3 of [RFC5322] describes a set of choices about whether (and how) to populate the Bcc field explicitly on Bcc'ed copies of the message, and in the copy stored in the sender's Sent folder.

* When separate copies are made for Bcced recipients, should each separate copy *also* be encrypted to the named recipients, or just to the designated Bcc recipient?

*When a copy is stored in the Sent folder, should that copy also be encrypted to Bcced recipients? (see also Section 9.1)

*When a message is encrypted, if there is a mechanism to include the certificates of the recipients, whose certificates should be included?

9.2.1. Simple Encryption with Bcc

Here is a simple approach that tries to minimize the total number of variants of the message created while leaving a coherent view of the message itself:

*No cryptographic payload contains any Bcc header field.

*The main copy of the message is signed and encrypted to all named recipients and to the sender. A copy of this message is also stored in the sender's Sent folder.

*Each Bcc recipient receives a distinct copy of the message, with an identical cryptographic payload, and the message is signed and encrypted to that specific recipient and all the named recipients. These copies are not stored in the sender's Sent folder.

*To the extent that spare certificates are included in the message, each generated copy of the message should include certificates for the sender and for each named recipient. Certificates for Bcc'ed recipients are not included in any message.

9.2.1.1. Rationale

The approach described in Section 9.2.1 aligns the list of cryptographic recipients as closely as possible with the set of named recipients, while still allowing a Bcced recipient to read their own copy, and to "Reply All" should they want to.

This should reduce user confusion on the receiving side: a recipient of such a message who naively looks at the user-facing headers from their own mailbox will have a good sense of what cryptographic treatments have been applied to the message. It also simplifies message composition and user experience: the message composer sees fields that match their expectations about what will happen to the message. Additionally, it may preserve the ability for a Bcc'ed recipient to retain their anonymity, should they need to offer the signed cryptographic payload to an outside party as proof of the original sender's intent without revealing their own identity.

9.3. Composing a Message to Heterogeneous Recipients

When sending a message that the user intends to be encrypted, it's possible that some recipients will be unable to receive an encrypted copy. For example, when Carol composes a message to Alice and Bob, Carol's MUA may be able to find a valid encryption-capable certificate for Alice, but none for Bob.

In this situation, there are four possible strategies, each of which has a negative impact on the experience of using encrypted mail. Carol's MUA can:

1. send encrypted to Alice and Bob, knowing that Bob will be unable to read the message.

2. send encrypted to Alice only, dropping Bob from the message recipient list.

3. send the message in the clear to both Alice and Bob.

4. send an encrypted copy of the message to Alice, and a cleartext copy to Bob.

Each of these strategies has different drawbacks.

The problem with approach 1 is that Bob will receive unreadable mail.

The problem with approach 2 is that Carol's MUA will not send the message to Bob, despite Carol asking it to.

The problem with approach 3 is that Carol's MUA will not encrypt the message, despite Carol asking it to.

Approach 4 has two problems:

* Carol's MUA will release a cleartext copy of the message, despite Carol asking it to send the message encrypted.

* If Alice wants to "reply all" to the message, she may not be able to find an encryption-capable certificate for Bob either. This puts Alice in an awkward and confusing position, one that users are unlikely to understand. In particular, if Alice's MUA is following the guidance about replies to encrypted messages in Section 5.4, having received an encrypted copy will make Alice's Reply buffer behave in an unusual fashion.

This is particularly problematic when the second recipient is not "Bob" but in fact a public mailing list or other visible archive, where messages are simply never encrypted.

Carol is unlikely to understand the subtleties and negative downstream interactions involved with approaches 1 and 4, so presenting the user with those choices is not advised.

The most understandable approach for a MUA with an active user is to ask the user (when they hit "send") to choose between approach 2 and approach 3. If the user declines to choose between 2 and 3, the MUA can drop them back to their message composition window and let them make alternate adjustments.

9.4. Message Transport Protocol Proxy: A Dangerous Implementation Choice

An implementor of end-to-end cryptographic protections may be tempted by a simple software design that piggybacks off of a mail protocol like SMTP, IMAP, or JMAP to handle message assembly and interpretation. In such an architecture, a naive MUA speaks something like a "standard" protocol like SMTP, IMAP, or JMAP to a local proxy, and the proxy handles signing and encryption (outbound), and decryption and verification (inbound) internally on behalf of the user. While such a "pluggable" architecture has the advantage that it is likely to be easy to apply to any mail user agent, it is problematic for the goals of end-to-end communication, especially in an existing cleartext ecosystem like e-mail, where any given message might be unsigned or signed, cleartext or encrypted. In particular:

* the user cannot easily and safely identify what protections any particular message has (including messages currently being composed), and

* the proxy itself is unaware of subtle nuances about the message that the MUA actually knows.

With a trustworthy and well-synchronized sidechannel or protocol extension between the MUA and the proxy, it is possible to deploy such an implementation safely, but the requirement for the sidechannel or extension eliminates the universal-deployability advantage of the scheme.

This section attempts to document some of the inherent risks involved with such an architecture.

9.4.1. Dangers of a Submission Proxy for Message Composition

When composing and sending a message, the act of applying cryptographic protections has subtleties that cannot be directly expressed in the SMTP protocol used by Submission [RFC6409], or in any other simple protocol that hands off a cleartext message for further processing.

For example, the sender cannot indicate via SMTP whether or not a given message *should* be encrypted (some messages, like those sent to a publicly archived mailing list, are pointless to encrypt), or select among multiple certificates for a recipient, if they exist (see Section 8.1.3).

Likewise, because such a proxy only interacts with the message when it is ready to be sent, it cannot indicate back to the user *during message composition* whether or not the message is able to be encrypted (that is, whether a valid certificate is available for each intended recipient). A message author may write an entirely different message if they know that it will be protected end-to-end; but without this knowledge, the author is obliged either to write text that they presume will be intercepted, or to risk revealing sensitive content.

Even without encryption, deciding whether to sign or not (and which certificate to sign with, if more than one exists) is another choice that the proxy is ill-equipped to make. The common message-signing techniques either render a message unreadable by any client that does not support cryptographic mail (i.e., PKCS7 signed-data), or appear as an attachment that can cause confusion to a naive recipient using a legacy client (i.e., multipart/signed). If the sender knows that the recipient will not check signatures, they may prefer to leave a cleartext message without a cryptographic signature at all.

Furthermore, handling encryption properly depends on the context of any given message, which cannot be expressed by the MUA to the Submission proxy. For example, decisions about how to handle encryption and quoted or attributed text may depend on the cryptographic status of the message that is being replied to (see Section 5.4).

Additionally, such a proxy would need to be capable of managing the user's own key and certificate (see Section 8.2). How will the implementation indicate to the user when their own certificate is near expiry, for example? How will any other error conditions be handled when communication with the user is needed?

While an extension to SMTP might be able to express all the necessary semantics that would allow a generic MUA to compose messages with standard cryptographic protections via a proxy, such an extension is beyond the scope of this document. FIXME: add a reference to JMAP cryptographic message composition work.

9.4.2. Dangers of an IMAP Proxy for Message Rendering

When receiving and rendering a message, the process of indicating to the user the cryptographic status of a message requires subtleties that are difficult to offer from a straightforwad IMAP (or POP, or JMAP) proxy.

One approach such a proxy could take is to remove all the Cryptographic Layers from a well-formed message, and to package a description of those layers into a special header field that the MUA can read. But this merely raises the question: what semantics need to be represented? For example:

- Was the message signed? If so, by whom? When?
- Should the details of the cryptographic algorithms used in any signatures found be indicated as well?
- Was the message encrypted? if so, to whom? What key was used to decrypt it?
- If both signed and encrypted, was the signing outside the encryption or inside?
- How should errant Cryptographic Layers (see Section 4.5) be dealt with?
- What cryptographic protections do the headers of the message have? (see [I-D.draft-ietf-lamps-header-protection])
- How are any errors or surprises communicated to the user?

If the proxy passes any of this cryptographic status to the client in an added header field, it must also ensure that no such header field is present on the messages it receives before processing it. If it were to allow such a header field through unmodified to any client that is willing to trust its contents, an attacker could spoof the field to make the user believe lies about the cryptographic status of the message. In order for a MUA to be confident in such a header field, then, it needs a guarantee from the proxy that any header it produces will be safe. How does the MUA reliably negogiate this guarantee with the proxy? If the proxy can no longer offer this guarantee, how will the MUA know that things have changed?

If such a proxy handles certificate discovery in inbound messages (see Section 8.1.1), it will also need to communicate the results of that discovery process to its corresponding proxy for message composition (see Section 9.4.1).

While an extension to IMAP (or POP, or JMAP) might be able to express all the necessary semantics that would allow a generic MUA to indicate standardized cryptographic message status, such an extension is beyond the scope of this document. FIXME: add a reference to JMAP cryptographic status work.

9.4.3. Who Controls the Proxy?

Finally, consider that the naive proxy deployment approach is risky precisely because of its opacity to the end user. Such a deployment could be placed anywhere in the stack, including on a machine that is not ultimately controlled by the end user, making it effectively a form of transport protection, rather than end-to-end protection.

A MUA explicitly under the control of the end user with thoughtful integration can offer UI/UX and security guarantees that a proxy cannot provide.

10. IANA Considerations

MAYBE: provide an indicator in the IANA header registry for which headers are "structural" and which are "user-facing"? This is probably unnecessary.

11. Security Considerations

This entire document addresses security considerations about end-to-end cryptographic protections for e-mail messages.

12. Acknowledgements

The set of constructs and recommendations in this document are derived from discussions with many different implementers, including Alexey Melnikov, Bernie Hoeneisen, Bjarni Rúnar Einarsson, David Bremner, Deb Cooley, Holger Krekel, Jameson Rollins, Jonathan Hammell, juga, Patrick Brunschwig, Santosh Chokhani, and Vincent Breitmoser.

13. References

13.1. Normative References

[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/info/rfc8551>.

[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>.

[RFC4289] Freed, N. and J. Klensin, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", BCP 13, RFC 4289, DOI 10.17487/RFC4289, December 2005, <https://www.rfc-editor.org/info/rfc4289>.

[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>.

[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/info/rfc8174>.

13.2. Informative References

[chrome-indicators] Schechter, E., "Evolving Chrome's security indicators", May 2018, <https://blog.chromium.org/2018/05/evolving-chromes-security-indicators.html>.

[EFAIL] "EFAIL", n.d., <https://efail.de>.

[ORACLE] Ising, F., Poddebniak, D., Kappert, T., Saatjohann, C., and S. Schinzel, "Content-Type: multipart/oracle Tapping into Format Oracles in Email End-to-End Encryption", n.d., <https://www.usenix.org/conference/usenixsecurity23/presentation/ising>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

[RFC3207] Hoffman, P., "SMTP Service Extension for Secure SMTP over Transport Layer Security", RFC 3207, DOI 10.17487/RFC3207, February 2002, <https://www.rfc-editor.org/info/rfc3207>.

[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>.

[RFC3274] Gutmann, P., "Compressed Data Content Type for Cryptographic Message Syntax (CMS)", RFC 3274, DOI 10.17487/RFC3274, June 2002, <https://www.rfc-editor.org/info/rfc3274>.

[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>.

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

[RFC5355] Stillman, M., Ed., Gopal, R., Guttman, E., Sengodan, S., and M. Holdrege, "Threats Introduced by Reliable Server Pooling (RSerPool) and Requirements for Security in Response to Threats", RFC 5355, DOI 10.17487/RFC5355, September 2008, <https://www.rfc-editor.org/info/rfc5355>.

[RFC6409] Gellens, R. and J. Klensin, "Message Submission for Mail", STD 72, RFC 6409, DOI 10.17487/RFC6409, November 2011, <https://www.rfc-editor.org/info/rfc6409>.

[I-D.draft-ietf-lamps-header-protection] Gillmor, D. K., Hoeneisen, B., and A. Melnikov, "Header Protection for S/MIME", Work in Progress, Internet-Draft, draft-ietf-lamps-header-protection-11, 24 January 2023, <https://www.ietf.org/archive/id/draft-ietf-lamps-header-protection-11.txt>.

[I-D.draft-bre-openpgp-samples-01] Einarsson, B. R., "juga", and D. K. Gillmor, "OpenPGP Example Keys and Certificates", Work in Progress, Internet-Draft, draft-bre-openpgp-samples-01, 20 December 2019, <https://www.ietf.org/archive/id/draft-bre-openpgp-samples-01.txt>.

[RFC9216] Gillmor, D. K., Ed., "S/MIME Example Keys and Certificates", RFC 9216, DOI 10.17487/RFC9216, April 2022, <https://www.rfc-editor.org/info/rfc9216>.

Appendix A. Test Vectors

FIXME: This document should contain examples of well-formed and malformed messages using cryptographic key material and certificates from [I-D.draft-bre-openpgp-samples-01] and [RFC9216].

It may also include example renderings of these messages.

A.1. Document History

A.1.1. Substantive changes from draft-ietf-...-04 to draft-ietf-...-05

*Adopt and update text about Bcc from draft-ietf-lamps-header-protection

*Add section about the dangers of an implementation based on a network protocol proxy

A.1.2. Substantive changes from draft-ietf-...-03 to draft-ietf-...-04

*Added reference to multipart/oracle attacks

*Clarified that "Structural Headers" are the same as RFC2045's "MIME Headers"

A.1.3. Substantive changes from draft-ietf-...-02 to draft-ietf-...-03

*Added section about mixed recipients

*Noted SMIMEA and OPENPGPKEY DNS RR cert discovery mechanisms

*Added more notes about simplified mental models

*More clarification on one-status-per-message

*Added guidance to defend against EFAIL

A.1.4. Substantive changes from draft-ietf-...-01 to draft-ietf-...-02

*Added definition of "user-facing" headers

A.1.5. Substantive changes from draft-ietf-...-00 to draft-ietf-...-01

*Added section about distinguishing Main Body Parts and Attachments

*Updated document considerations section, including reference to auto-built editor's copy

A.1.6. Substantive changes from draft-dkg-...-01 to draft-ietf-...-00

*WG adopted draft

*moved Document History and Document Considerations sections to end of appendix, to avoid section renumbering when removed

A.1.7. Substantive changes from draft-dkg-...-00 to draft-dkg-...-01

*consideration of success/failure indicators for usability

*clarify extendedKeyUsage and keyUsage algorithm-specific details

*initial section on certificate management

*added more TODO items

Author's Address

Daniel Kahn Gillmor (editor)
American Civil Liberties Union
125 Broad St.
New York, NY, 10004
United States of America

Email: dkg@fifthhorseman.net