Stream:	Internet Engineering Task Force (IETF)		
RFC:	9787		
Category:	Informational		
Published:	August 2025		
ISSN:	2070-1721		
Authors:	D. K. Gillmor, Ed. *ACLU*	A. Melnikov, Ed. *Isode Ltd*	B. Hoeneisen, Ed. *pEp Project*

RFC 9787
Guidance on End-to-End Email Security

Abstract

End-to-end cryptographic protections for email 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 (MUA) implementers to help mitigate those risks and to make end-to-end email simple and secure for the end user. It provides a useful set of vocabulary as well as recommendations to avoid common failures. It also identifies a number of currently unsolved usability and interoperability problems.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9787.

Copyright Notice

Table of Contents

1. Introduction

End-to-end email security using S/MIME [RFC8551] and PGP/MIME (Pretty Good Privacy with MIME) [RFC3156] cryptographic standards can provide integrity, authentication, and confidentiality to MIME [RFC4289] email messages.

However, there are many ways that an MUA handling such a message can misinterpret or accidentally break these security guarantees. For example, as described in [EFAIL], the "Direct Exfiltration" attack leaks cleartext due to an attack that splices existing ciphertext into a new message, which is then handled optimistically (and wrongly) by many MUAs.

A MUA 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 MUA 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.

This document offers guidance to the implementer of a MUA that provides these cryptographic protections, whether for composing, interpreting, rendering, or replying to a message. An implementation that follows this guidance will provide its users with stronger and easier-to-understand security properties and will also offer more reliable interoperability for messages exchanged with other implementations.

In Appendix A, this document also identifies a number of interoperability and usability concerns for end-to-end cryptographic email that have no current broadly accepted technical standard for resolution. One major area not covered in this document is the acquisition and long-term maintenance of cryptographic identity information and metadata across multiple MUAs controlled by the same user.

1.1. Terminology

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.

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

- The *Mail User Agent (MUA)* is an email client.
- *Protection* of message data refers to cryptographic encryption and/or signatures, providing confidentiality, authenticity, and/or integrity.
- *Cryptographic Layer*, *Cryptographic Envelope*, *Cryptographic Payload*, *Cryptographic Summary*, and *Errant Cryptographic Layer* are defined in Section 4.
- A *well-formed* email message with cryptographic protection has both a *Cryptographic Envelope* and a *Cryptographic Payload*.
- *Structural Header Fields* are documented in Section 1.1.1.
- *Non-Structural Header Fields* are header fields that are not Structural Header Fields.
- *User-Facing Header Fields* are documented in Section 1.1.2.
- A *Main Body Part* is any part of a message that is typically rendered to the user as the message itself (not "as an attachment"). See Section 7.1.

This document contains extensive discussion about end-to-end cryptographic protections in email while acknowledging that many MUAs have no capabilities for end-to-end cryptographic protections at all. We divide MUAs into three distinct profiles:

- A *non-cryptographic* MUA has no capabilities for end-to-end cryptographic protections.
- A *conformant* MUA follows the guidance in this document when dealing with end-to-end cryptographic protections.
- A *legacy* MUA has capabilities for end-to-end cryptographic protections but does not adhere to the guidance in this document.

At the time of the writing of this document, most MUAs with cryptographic protections are legacy MUAs.

1.1.1. Structural Header Fields

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

These header fields 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 Header Fields

Of all the header fields that an email message may contain, only a small subset are typically presented directly to the user. This document refers to them as User-Facing Header Fields. Typically, User-Facing Header Fields are:

- `Subject`
- `From`
- `To`
- `Cc`
- `Date`
- `Reply-To`
- `Followup-To`
- `Sender`
- `Resent-From`
- `Resent-To`
- `Resent-Cc`
- `Resent-Date`
- `Resent-Sender`

The above list contains the header fields most often presented directly to the user who views a message, though an MUA may also decide to treat any other header field as "User-Facing". Of course, many of these header fields are entirely absent from any given message, and an absent header field is not presented to the user at all.

Note that the resending header fields (those beginning with `Resent-`) are typically only added by an intervening MUA (see Section 3.6.6 of [RFC5322] and Section 9.8 of this document). As such, though they may in some cases be presented to the user, they will typically not bear any end-to-end cryptographic protection (even if the original header fields of a message are protected; see Section 9.3), because they are unknown to the original sender.

Other header fields 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, or the `List-*` and `Archived-At` header fields added by mailing lists may cause additional buttons to be displayed during rendering), 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 interfere with communication should be avoided where possible.

Furthermore, it is a near certainty 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 email 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 that 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 details about mental models and cryptographic status.

> (It is of course possible that a message forwarded as a MIME attachment could have its own cryptographic status while still being a message subpart, but that status should be distinct from the status of the enclosing message.)

2.2. Email 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.

Email offers a few distinctions from these other systems, most notably features like:

Ubiquity: Most correspondents will have an email address, while not everyone is present on the various non-email messaging services, particularly those that have reliable end-to-end cryptographic protections.

Federation: Interaction between users on distinct domains who have not agreed on a common communications provider is still possible.

User Control: The user can interact with the email system using an 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 email is likely using email instead of other systems because of the distinctions outlined above. When adding end-to-end cryptographic protection to an email endpoint, care should be taken not to negate any of the distinct features of email as a whole. If these features are violated to provide end-to-end cryptographic protection, the user may just as well choose one of the other systems that don't have the drawbacks that email has. Implementers should try to provide end-to-end protections that retain the familiar experience of email itself.

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

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 email.

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. Starting 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 an MUA first enables end-to-end cryptographic protection, it's likely that they will want to see which messages *have* protection (that is, the security indicators amenable to a conformant MUA as of 2025 are most likely to be comparable to those of a pre-2018 web browser). But a user whose private email communications with a given correspondent, or within a given domain, are known to be entirely end-to-end protected might instead want to know which messages do *not* have the expected protections.

Note also that some messages may be 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 offering a simple user experience should be deliberate and judicious about the expectations their interface assumes that the user has in a given context. See Appendix A.9 for future work.

3. Types of Protection

A given message might be:

- signed,
- encrypted,
- both signed and encrypted, or
- none of the above.

Given that many email 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 email message actually does have end-to-end cryptographic protections, those protections need to be visible and comprehensible to the end users: both the composing user and the recipient who views the rendered message. If either user is unaware of the protections, then they do not effectively 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 recipient could be that a message is in one of three normal states, which align with the only reasonable choices for message composition:

- Unprotected
- Verified (has a valid signature from the apparent sender of the message)
- Confidential (encrypted with a valid signature from the apparent sender of the message)

However, one error state exists for rendered mail that does not correspond to a reasonable choice for message composition:

- Encrypted But Unverified (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. However, this is not the state of the global email ecosystem when this document was written, since some legacy MUAs permit composing encrypted-but-unsigned mail (see Appendix A.9 for possible future guidance).

Alternately, an MUA may prefer to represent the state of an 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, 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) unless the MUA is providing the user with debugging information.

At the time this document was written, the global email ecosystem contains a heterogeneous mix of legacy and non-cryptographic MUAs. In such an ecosystem, a conformant MUA may instead prefer to represent "Verified" and "Encrypted" as orthogonal states of any given rendered message. While this model does not precisely match the choice a user makes when composing a message, it may align more with the reality of the range of messages they encounter as a recipient.

3.2. One Cryptographic Status per Message

Some MUAs may attempt to generate multiple copies of a given email 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 composer 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 9.6 for how this can go wrong.

For comprehensibility, a conformant 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 due to ease of implementation and deployment. These opportunistic transport protections are orthogonal to the end-to-end protections described in this document.

To the extent that opportunistic message protections are made visible to the user for a given copy of a message, a conformant 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 (see [RFC7435]).

The user needs a single clear, simple, and correct indication about the end-to-end cryptographic status of any given message. See Section 4.6 for more details.

4. Cryptographic MIME Message Structure

Implementations use the structure of an email message to establish (when composing) and understand (when rendering) 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, "╧" (BOX DRAWINGS UP SINGLE AND HORIZONTAL DOUBLE, U+2567) indicates "decrypts to" and "┴" (BOX DRAWINGS LIGHT UP AND HORIZONTAL, U+2534) 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, PKCS #7 enveloped-data, and PKCS #7 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 PKCS #7 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 PKCS #7 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 PKCS #7 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 structures and very similar confidentiality 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. PKCS #7 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:

- confidentiality (via a Symmetrically Encrypted Data Packet, see Section 5.7 of [RFC9580]; an MUA **MUST NOT** generate this form due to ciphertext malleability),
- confidentiality and integrity (via a Symmetrically Encrypted and Integrity Protected Data Packet (SEIPD); see Section 5.13 of [RFC9580]), or
- confidentiality, integrity, and authenticity all together (by including an OpenPGP Signature Packet, see Section 5.2 of [RFC9580], within the SEIPD).

4.2. Cryptographic Envelope

The Cryptographic Envelope is the largest contiguous set of Cryptographic Layers of an email 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).

Also note 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 [RFC9580] Encrypted Message within the `application/octet-stream` part contains a [RFC9580] 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` and `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 Cryptographic Layers

Due to confusion, malice, message forwarding, 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 **MUST 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 rewrap a well-formed signed message before resending to add a footer, resulting in the following structure seen by recipients of the email:

```
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 email messages with this structure, because a legacy MUA may present the data in part `L` as though it were part of `J`, though they have different cryptographic properties. In particular, if the user believes that the entire 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.

See also Section 6.2.1.1 for guidance on how a rendering MUA might deal with this common pattern.

4.5.2. A Baroque Example

Consider a message with the following overcomplicated structure:

```
M └─ multipart/encrypted
N   ├─ application/pgp-encrypted
O   └─ application/octet-stream
P    ═ (decrypts to)
Q     └─ multipart/signed
R       ├─ multipart/mixed
S       │  ├─ multipart/signed
T       │  │  ├─ text/plain
U       │  │  └─ application/pgp-signature
V       │  └─ text/plain
W       └─ application/pgp-signature
```

The three 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` and `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`.

4.6. Cryptographic Summary

The cryptographic status of an email message with end-to-end cryptographic protections is known as the Cryptographic Summary. A reasonable, simple Cryptographic Summary is derived from the aggregate properties of the layers in the Cryptographic Envelope. This is a conceptual tool and a feature that an MUA can use to guide behavior and user experience, but it is not necessarily always directly exposed in any given user interface. See Section 6.1 for guidance and considerations about rendering the Cryptographic Summary to the user.

5. Message Composition

This section describes the ideal composition of an email 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 describes the steps that an MUA should use to compose a cryptographically protected message, such that it has a proper Cryptographic Envelope and Payload.

The inputs to the algorithm are:

`origbody`: The 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 Header Fields present (see Section 1.1.1).

`origheaders`: The intended Non-Structural Header Fields 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:

1. Apply `crypto` to MIME part `origbody`, producing MIME tree `output`.
2. For each header field name and value `(h,v)` in `origheaders`:
   a. Add header field `h` to `output` with value `v`.
3. Return `output`.

This is the classic algorithm. It only protects the Structural Header Fields of the message body and leaves Non-Structural (including User-Facing) Header Fields unprotected.

Therefore, a conformant MUA **MUST** implement Header Protection as described in [RFC9788] (see Section 9.3).

5.2. Encryption Outside, Signature Inside

An email message that is both signed and encrypted is signed *inside* the encryption and not the other way around. For example, when crafting a signed-and-encrypted message using a simple Cryptographic Envelope of a single layer (Section 4.4.1) with PGP/MIME, the OpenPGP Encrypted Message object should contain an OpenPGP Signed Message. Likewise, when using a multilayer Cryptographic Envelope (Section 4.4.2), the outer layer should be an encryption layer and the inner layer should be a signing layer.

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 conformant MUA **MUST 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 email, there are four options for applying end-to-end cryptographic protections:

- Unprotected: In some cases, offering any end-to-end cryptographic protection is harmful: It may confuse the recipient and offer no benefit.
- Signed-only: In other cases, signing a message is useful (authenticity and integrity are desirable), but encryption is either impossible (for example, if the composer does not know how to encrypt to all recipients) or meaningless (for example, an email message to a mailing list that is intended to be published to a public archive).
- Signed-and-encrypted: In other cases, full end-to-end confidentiality, authenticity, and integrity are desirable.
- Encrypted-only: There is no common use case for generating an email message with end-to-end confidentiality but without authenticity or integrity.
  - Additionally, some encryption schemes are malleable. This allows an attacker to tamper with the plaintext by modifying the ciphertext (i.e., without decrypting it). One way to prevent this is to authenticate the ciphertext, e.g., using signatures, Message Authentication Codes (MACs), or Authenticated Encryption with Associated Data (AEAD) schemes. See the Cipher Block Chaining (CBC) / Cipher FeedBack (CFB) gadget attack in [EFAIL].

A conformant MUA should keep its message composition interface simple. When presenting the user with a choice of cryptographic protection, it **MUST** offer no more than three choices:

1. No end-to-end cryptographic protection
2. Verified (signed only)
3. Confidential (signed and encrypted)

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

A common anti-pattern among legacy MUAs is offering two boolean choices: one to toggle signing and another to toggle encryption. This creates usability hurdles:

- A user who wants to compose a signed-and-encrypted message will have to click two buttons instead of one.
- A user who clicks "Encrypt" but neglects to click "Signed" may not understand that they are creating a message that cannot be authenticated by the recipient.

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, and 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, unless the user indicates that they are deliberately including previously confidential content in a non-confidential 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 composers to create email messages with well-formed, end-to-end cryptographic protection, rendering 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 rendering MUA evaluates and assembles the cryptographic properties of the Cryptographic Envelope into a Cryptographic Summary and displays that status to the user in a secure and 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 rendered message itself. By analogy, consider the "lock" icon in the address bar of the web browser: Regardless of the content of the webpage, the lock icon will only be displayed when the transport to the web server is adequately secured.

Aside from this Cryptographic Summary, the message itself **MUST** be rendered as though the Cryptographic Payload is the body of the message. The Cryptographic Layers themselves **SHOULD NOT** be rendered as distinct objects unless the MUA is providing the user with debugging information.

6.2. Errant Cryptographic Layers

If an incoming message has any Errant Cryptographic Layers, a conformant interpreting MUA **MUST** 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, a conformant MUA **MUST NOT** indicate in the Cryptographic Summary that the message is signed. It **MUST** indicate that the message is Unprotected.

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 conformant 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 field) 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 Verified 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 positive Cryptographic Summary but shows the footer:

```
Cryptographic Protections: Unprotected
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 as Verified but hides the footer:

```
Cryptographic Protections: Verified [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 have access to, so a conformant MUA in this situation **SHOULD** decrypt the part.

However, in this case, a conformant 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 it **MUST 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, unless quoted and attributed text is not included in the reply, as noted in Section 5.4.

When composing a reply to a message with an Errant Cryptographic Layer, a conformant MUA **MUST NOT** quote or attribute text derived from the decryption of any Errant Cryptographic Layer. This will typically mean either leaving the ciphertext itself in the generated reply message or simply not generating any quoted or attributed text at all. This offers protection against the reply-based attacks described in [REPLY].

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 [RFC9580]) ciphertext 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 conformant 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 Unprotected. See [SPOOFING] for examples of spoofed message signatures that rely on permissive legacy clients that are willing to validate signatures in poorly structured messages.

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, a conformant MUA **MAY** fully decline to decrypt the data.

During message rendering, if a conformant MUA attempts decryption of such a non-MIME encrypted section of an email, it **MUST** synthesize a separate MIME part to contain only the decrypted data and it **MUST 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, a conformant MUA **MUST NOT** decrypt 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 [REPLY].

6.3. Forwarded Messages with Cryptographic Protection

An incoming email message may include an attached forwarded message, typically as a MIME subpart with `Content-Type: message/rfc822` [RFC5322] or `Content-Type: message/global` [RFC6532].

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 Errant Cryptographic Layers with respect to the surrounding message -- they are layers that only apply to the forwarded message itself.

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

A conformant 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 by either the enclosing message or the forwarded message.

6.4. Signature Failures

A cryptographic signature may fail in multiple ways. A conformant rendering MUA that discovers a failed signature treats the message as though the signature did not exist. This is similar to the standard guidance for failed DomainKeys Identified Mail (DKIM) signatures (see Section 6.1 of [RFC6376]).

A conformant MUA **MUST NOT** render a message with a failed signature as more dangerous or more dubious than a comparable message without any signature at all. In both cases, the Cryptographic Summary should be Unprotected.

A conformant MUA that encounters a signed-and-encrypted message where the signature is invalid **SHOULD** treat the message the same way that it would treat a message that is encryption-only, unless the MUA is providing the user with debugging information.

These are 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 deprecated digest algorithm, or was made by a weak key, e.g., 1024-bit RSA); note that this requires the rendering MUA to have an explicit policy about what cryptographic primitives are acceptable.
- The signature is made by a certificate that the rendering MUA does not have access to.
- The certificate used to verify the signature was revoked.
- The certificate used to verify the signature was expired at the time that the signature was made.
- The certificate used to verify 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 email address found in the `From` or `Sender` header field.)
- The certificate used to verify the signature was not issued by an authority that the MUA user is willing to rely on for certifying the sender's email address, and the user has no other reasonable indication that the certificate belongs to the sender's email address.
- The signature indicates that it was made at a time much before or much after the date of the message itself.
- The signature covers a message that depends on an external subresource that might change (see Section 9.9).

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.

6.5. Weak Encryption

Sometimes, an MUA might encounter a message with a well-formed Cryptographic Envelope that uses a form of encryption with substantial known flaws. For example, a PGP/MIME message might use a Symmetrically Encrypted Data Packet, which is known to have malleable ciphertext (see Section 5.7 of [RFC9580]). As another example, an S/MIME message may use an `enveloped-data` MIME part with a `contentEncryptionAlgorithm` of `rc2-cbc` with `rc2ParameterVersion` of 160, meaning a 40-bit key (see Section 5.2 of [RFC3370]), which is widely considered breakable via brute force with moderate hardware investment in 2024. As cryptanalysis and hardware capacities advance, an implementation may widen the scope of what encryption mechanisms are considered weak.

A rendering MUA **MUST** warn the user that such a message has a known weakness. The rendering MUA **MAY** fully decline to decrypt such a message. If it decides to decrypt a message with a weak encryption layer, it **MUST NOT** indicate in the message's Cryptographic Summary that the message was encrypted, as the confidentiality of the message is suspect. This is similar to the approach taken in Section 6.2.2 for messages with an Errant Encryption Layer.

Like the Errant Encryption Layer situation, there is an asymmetry between rendering and replying to a message with weak encryption. The guidance in Section 6.2.2.1 should be followed when replying to a message with weak encryption as well.

A rendering MUA **MAY** also treat historically archived messages with weak encryption differently than modern messages. For example, if an encryption algorithm was known to be weak in 2005, then a message that appears to have been encrypted with that algorithm in 1995 might be decrypted with a warning, as an archival service. But a message that appears to have been encrypted with that same weak algorithm in 2015 might be quarantined as a likely attack.

There are several possible ways to distinguish a historical message from a modern one, including:

- the message timestamp (e.g., the `Date` header field)
- the time the message was first observed on the network (e.g., delivery of a new message via IMAP from a mailbox that had recently checked)
- the timestamp in any signature observed in the message
- the message structure (a message composed using protocol elements that were invented after the encryption algorithm was known as weak is more likely to be an attack than a legitimate archival message)

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 email 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 rendering 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 `text` (e.g., `text/plain` or `text/html`) as a primary content type 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.

An MUA **MAY** let the user select a preferred MIME type for rendering 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 rendering MUA, a conformant composing MUA should consider that multiple parts might be rendered as the Main Body Part when the message is ultimately viewed. In particular, the composing MUA should ensure that any part likely to be viewed as the Main Body Part has the same semantic content as any other such part.

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 email 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 the Main Body Parts, it can identify the likely parts by following the same routine as a rendering 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`.

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`.

When generating 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) and will not be handled by other MUAs as intended.

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.
- The divisions between the different attachments are visible to operators of any mail transport agent (MTA) that handles the message, potentially resulting in a metadata leak. For example, the MTA operator may learn the number of attachments and the size of each attachment.

These messages are unlikely to be usefully interoperable without additional standardization work (see Appendix A.12).

7.3. MIME Part Examples

Consider a common message with the following 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 field, it is potentially ambiguous whether it is an attachment or not. If the composer prefers a specific behavior, it should explicitly set the `Content-Disposition` header field on part S to either `inline` or `attachment` as guidance to the rendering MUA.

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 still comprise 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. Detailed guidance about how to do this is beyond the scope of this document, but future revisions may bring it into scope (see Appendix A.3).

8.1.1. Peer Certificate Selection

When composing an encrypted message, the MUA needs to select an encryption-capable certificate for each recipient.

To select such a certificate for a given destination email 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 (for all ASCII email addresses) or SmtpUTF8Mailbox (for Internationalized Email Addresses) whose contents match the destination email address. In particular, for rfc822Name, the `local-part` of the two addresses should be an exact bytewise match, and the domain parts of the two addresses should be matched by ensuring label equivalence across the full domain name, as described in Section 2.3.2.4 of [RFC5890]. Comparison with SmtpUTF8Mailbox is specified in Section 5 of [RFC9598].
- The algorithm OID in the certificate's SubjectPublicKeyInfo (SPKI) is known to the MUA and capable of encryption. Examples include:
  - rsaEncryption (OID 1.2.840.113549.1.1.1), with the keyUsage (OID 2.5.29.15) extension present and the "key encipherment" bit (value 32) set.
  - curveX25519 (OID 1.3.101.110), with the keyUsage extension present and the "key agreement" bit (value 8) set.
- If extendedKeyUsage (OID 2.5.29.37) is present, it contains at least one of the following OIDs: email protection (OID 1.3.6.1.5.5.7.3.4), anyExtendedKeyUsage (OID 2.5.29.37.0).

A conformant MUA may include more considerations when selecting a peer certificate as well; see Appendix A.3.4 for examples.

8.2. Local Certificates

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

While some basic guidance is offered here, it is beyond the scope of this document to describe all possible relevant guidance for local certificate and key material handling. See Appendix A.4 for suggestions of guidance that a future version might bring into scope.

8.2.1. Getting Certificates for the User

If a conformant MUA does not have a certificate or secret key for the user, it should help the user to generate, acquire, or import them with minimum difficulty.

8.2.1.1. User Certificates for S/MIME

For S/MIME, the user **SHOULD** have both a signing-capable certificate and an encryption-capable certificate (and the corresponding secret keys). Using the same cryptographic key material for multiple algorithms (i.e., for both encryption and signing) has been the source of vulnerabilities in other (non-email) contexts (e.g., [DROWN] and [IKE]). The simplest way to avoid any

comparable risk is to use distinct key material for each cryptographic algorithm. A conformant MUA that generates S/MIME certificates for the user **MUST** generate distinct S/MIME certificates to avoid possible cross-protocol key misuse: one for encryption and another for signing.

The simplest option for an S/MIME-capable MUA is for the MUA to permit the user to import a PKCS #12 [RFC7292] object that is expected to contain secret key material, end entity certificates for the user, and intermediate certification authority (CA) certificates that permit chaining from the end entity certificates to widely accepted trust anchors. A conformant MUA that imports such a PKCS #12 bundle **SHOULD** warn the user if the bundle contains an S/MIME certificate and corresponding secret key where the same secret key is used for both encryption and signing.

An S/MIME-capable MUA that has access to user certificates and their corresponding secret key material should also offer the ability to export those objects into a well-formed PKCS #12 object that could be imported into another MUA operated by the same user.

Manual handling of PKCS #12 objects is challenging for most users. Producing the initial PKCS #12 object typically can only be done with the aid of a CA via non-standardized, labor-intensive, and error-prone procedures that most users do not understand. Furthermore, manual export and import incurs ongoing labor (for example, before certificate expiration) by the user, which most users are unprepared to do (see Section 8.2.2).

A better approach is for the MUA to integrate some form of automated certificate issuance procedure, for example, by using the Automatic Certificate Management Environment (ACME) protocol for end user S/MIME certificates [RFC8823].

Another possible approach is integration with a cryptographic hardware token or smart card that can provide certificates and permit the use of isolated secret key material, for example, see [PKCS11], though this approach delegates the complexity of acquiring and managing certificates to management of the hardware token itself (see Appendix A.4.2).

See also [CERT-BEST-PRACTICE] for more recommendations about managing user certificates.

8.2.1.2. User Certificates for PGP/MIME

As distinct from S/MIME, OpenPGP [RFC9580] has a different set of common practices. For one thing, a single OpenPGP certificate can contain both a signing-capable key and a distinct encryption-capable key, so only one certificate is needed for an email user of OpenPGP as long as the certificate has distinct key material for the different purposes.

Furthermore, a single OpenPGP certificate **MAY** only be self-signed, so the MUA can generate such a certificate entirely on its own.

An OpenPGP-capable MUA should have the ability to import and export OpenPGP Transferable Secret Keys (see Section 10.2 of [RFC9580]) to enable manual transfer of user certificates and secret key material between multiple MUAs controlled by the user.

Since an OpenPGP certificate **MAY** be certified by third parties (whether formal CAs or merely other well-connected peers), the MUA **SHOULD** offer affordances to help the user acquire and merge third-party certifications on their certificate. When doing this, the MUA should prioritize third-party certifications from entities that the user's peers are likely to know about and be willing to rely on.

Since an OpenPGP certificate can grow arbitrarily large with third-party certifications, the MUA should assist the user in pruning it to ensure that it remains a reasonable size when transmitting it to other parties.

8.2.1.3. Generate Secret Key Material Locally

Regardless of the protocol used (S/MIME or PGP), when producing certificates for the end user, the MUA **SHOULD** ensure that it has generated secret key material locally and **MUST NOT** accept secret key material from an untrusted external party as the basis for the user's certificate. For example, a user who trusts their system administrator not to compromise their MUA may accept secret key material generated by the system administrator but probably should not accept secret key material generated by an unaffiliated online web service.

An MUA that accepts secret key material from a third party cannot prevent that third party from retaining this material. A third party with this level of access could decrypt messages intended to be confidential for the user or could forge messages that would appear to come from the user.

8.2.2. Local Certificate Maintenance

In the context of a single email account managed by an MUA, where that email account is expected to be able to use end-to-end cryptographic protections, the MUA **SHOULD** warn the user (or proactively fix the problem) when/if:

- For S/MIME, the user's own certificate set for the account does not include a valid, unexpired encryption-capable X.509 certificate and a valid, unexpired signature-capable X.509 certificate.
- For PGP/MIME, the user's own certificate does not include a valid, unexpired signing-capable key/subkey and a valid, unexpired encryption-capable key/subkey.
- Any of the user's own certificates for the account:
  - are due to expire in the next month (or at some other reasonable cadence).
  - do not match the email address associated with the account in question.
- Any of the user's own S/MIME certificates for the account:
  - do not have a keyUsage extension.
  - do not contain an extendedKeyUsage extension.
  - would be considered invalid by the MUA for any other reason if it were a peer certificate.

An MUA that takes active steps to fix any of these problems before they arise is even more usable than one that just issues warnings, but guidance on how to do active certificate maintenance is beyond the scope of this document (see Appendix A.4.3).

If the MUA does find any of these issues and chooses to warn the user, it should use one aggregate warning with simple language that describes how the certificates might not be acceptable for other people and recommend a course of action that the user can take to remedy the problem.

8.2.3. Shipping Certificates in Outbound Messages

When composing mail, a conformant MUA **SHOULD** include copies of the user's own certificates (and potentially other certificates) in each message to facilitate future communication, unless it has specific knowledge that the other parties involved already know the relevant keys (for example, if it is mail between members within a domain that has a synchronized and up-to-date certificate directory).

The mechanism for including these certificates, and which certificates to include in the message, are protocol specific.

8.2.3.1. Shipping Certificates in S/MIME Messages

In any S/MIME SignedData object, certificates can be shipped in the "certificates" member. In any S/MIME EnvelopedData object, certificates can be shipped in the "originatorInfo.certs" member.

When a single S/MIME-protected email message is signed-and-encrypted, it is usually sufficient to ship all the relevant certificates in the inner SignedData object's "certificates" member.

The S/MIME certificates shipped in such a message **SHOULD** include:

- the user's own S/MIME signing certificate, so that signature on the current message can be validated.
- the user's own S/MIME encryption-capable certificate, so that the recipient can reply in encrypted form.
- on an encrypted message to multiple recipients, the encryption-capable peer certificates of the other recipients, so that any recipient can easily "reply all" without needing to search for certificates.
- any intermediate CA certificates needed to chain all of the above to a widely trusted set of root authorities.

8.2.3.2. Shipping Certificates in PGP/MIME Messages

PGP/MIME does not have a single specific standard location for shipping certificates.

Some MUAs ship relevant OpenPGP certificates in a single MIME leaf of Content-Type "application/pgp-keys". When such a message has cryptographic protections, to ensure that the message is well-formed, this kind of MIME part **SHOULD** be a leaf of the Cryptographic Payload and not outside of it. One problem with this approach is that it appears to recipients with non-cryptographic MUAs as an "attachment", which can lead to confusion if the user does not know how to use it.

Other implementations ship relevant OpenPGP certificates in "Autocrypt" or "Autocrypt-Gossip" message header fields (see [AUTOCRYPT]). To ensure that those header fields receive the same cryptographic authenticity as the rest of the message, these header fields **SHOULD** be protected as described in [RFC9788].

The OpenPGP certificates shipped in such a message **SHOULD** include:

- the user's own OpenPGP certificate, capable of both signing and encryption, so that the user can validate the message's signature and can encrypt future messages.
- on an encrypted message to multiple recipients, the OpenPGP certificates of the other recipients, so that any recipient can easily "reply all" without needing to search for certificates.

9. Common Pitfalls and Guidelines

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

9.1. Reading Sent Messages

When sending 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 reread 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 trade-offs:

- 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 resubmit 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, in any attempt to resend the message, the cryptographic transformation needs to be reapplied before sending or else the message contents will leak upon resend. A conformant MUA **SHOULD NOT** store a cleartext copy in the Sent folder unless it knows that the Sent folder cannot be read by an attacker. For example, if end-to-end confidentiality is desired, then storing the cleartext in an IMAP folder where a potentially adversarial server can read it defeats the purpose.

- A final option is that the MUA can store a copy of the message's encryption session key. Standard email 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. Reading Encrypted Messages after Certificate Expiration

When encrypting a message, the composing MUA should decline to encrypt to an expired certificate (see Section 8.1.1). But when decrypting a message, as long as the viewing MUA has access to the appropriate secret key material, it should permit decryption of the message, even if the associated certificate is expired. That is, the rendering MUA should not prevent the user from reading a message that they have access to merely due to an expired encryption certificate.

The rendering MUA may warn the user when decrypting a message that appears to have been encrypted to an encryption-capable certificate that was expired at the time of encryption (e.g., based on the `Date` header field of the message or the timestamp in the cryptographic signature) but otherwise should not complain.

The primary goal of certificate expiration is to facilitate rotation of secret key material, so that secret key material does not need to be retained indefinitely. Certificate expiration permits the user to destroy an older secret key if access to the messages encrypted to it is no longer necessary (see also Appendix A.10).

9.3. Unprotected Message Header Fields

Many legacy cryptographically aware MUAs only apply cryptographic protections to the body of the message but leave the header fields unprotected. This gives rise to vulnerabilities like information leakage (e.g., the Subject line is visible to a passive intermediary) or message tampering (e.g., the Subject line is replaced, effectively changing the semantics of a signed message). These are not only security vulnerabilities but also usability problems, because the distinction between what is part of the header section and what is part of the body of a message is unclear to many end users and requires a more complex mental model than is necessary. Useful security comes from alignment between simple mental models and tooling.

To avoid these concerns, a conformant MUA **MUST** implement Header Protection as described in [RFC9788].

Note that some message header fields, such as `List-*`, `Archived-At`, and `Resent-*`, are typically added by an intervening MUA (see Section 9.8), not by one of the classic "ends" of an end-to-end email exchange. A rendering MUA may choose to consider the contents of these header fields on an end-to-end protected message as markers added during message transit, even if they are not covered by the end-to-end cryptographic protection.

9.4. 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 header fields.

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.

- Should the `Bcc` header field be populated explicitly on `Bcc`'ed copies of the message and in the copy stored in the sender's `Sent` folder? See Section 3.6.3 of [RFC5322] for a set of choices.
- When separate copies are made for `Bcc`'ed 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 `Bcc`'ed 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?

9.4.1. 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 (that is, the cleartext is identical), 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.
- Any `Bcc`'ed recipient **MUST NOT** be taken into consideration when determining which certificates to include in the message. In particular, certificates for `Bcc`'ed recipients **MUST NOT** included in any message.

9.4.1.1. Rationale

The approach described in Section 9.4.1 aligns the list of cryptographic recipients as closely as possible with the set of named recipients while still allowing a `Bcc`'ed 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 Header Fields 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.

9.5. Draft Messages

When composing a message, most MUAs will save a copy of the as-yet-unsent message to a "Drafts" folder. If that folder is itself stored somewhere not under the user's control (e.g., an IMAP mailbox), it would be a mistake to store the draft message in the clear, because its contents could leak.

This is the case even if the message is ultimately sent deliberately in the clear. During message composition, the MUA does not know whether the message is intended to be sent encrypted or not. For example, just before sending, the user could decide to encrypt the message, and the MUA would have had no way of knowing.

The MUA **SHOULD** encrypt all draft messages, unless it has explicit knowledge that the message will not be encrypted when sent or that the Drafts folder cannot be read by an attacker. For example, if end-to-end confidentiality is desired, then storing a cleartext draft in an IMAP folder where a potentially adversarial server can read it defeats the purpose.

Furthermore, when encrypting a draft message, the message draft **MUST** only be encrypted to the user's own certificate or to some equivalent secret key that only the user possesses. A draft message encrypted in this way can be decrypted when the user wants to resume composing the message but cannot be read by anyone else, including a potential intended recipient. Note that a draft message encrypted in this way will only be resumable by another MUA attached to the same mailbox if that other MUA has access to the user's decryption-capable secret key. This shared access to key material is also likely necessary for useful interoperability but is beyond the scope of this document (see Appendix A.4.1).

A conformant MUA **MUST NOT** sign a message draft with the user's normal signing key, because creating a non-repudiable signature implies a commitment from the sender. If a signed draft message were to leak to the user's "Drafts" folder on some untrustworthy server, the server operator could claim that the user had committed to something that they had not yet decided to commit to. If draft signing is intended for cryptographic coordination between multiple MUAs of the same user, it should be negotiated with a different key (but see Appendix A.4.1).

The message should only be encrypted to its recipients upon actually sending the message. No reasonable user expects their message's intended recipients to be able to read a message that is not yet complete.

9.6. Composing a Message to Heterogeneous Recipients

When composing a message that the user intends to be encrypted, it's possible that some recipients will be unable to view 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:

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

See also further discussion of these scenarios in [CLEARTEXT-COPY].

9.7. Message Transport Protocol Proxy: A Dangerous Implementation Choice

An implementer of end-to-end cryptographic protections may be tempted by a simple software design that piggybacks off of a mail protocol, like SMTP Submission [RFC6409], IMAP [RFC9051], or JSON Meta Application Protocol (JMAP) [RFC8621], 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 of likely being easy to apply to any MUA, it is problematic for the goals of end-to-end communication, especially in an existing cleartext ecosystem like email, 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 side channel or protocol extension between the MUA and the proxy, it is possible to deploy such an implementation safely, but the requirement for the side channel or extension eliminates the universal deployability advantage of the scheme.

Similar concerns apply to any implementation bound by an API that operates on message objects alone, without any additional contextual parameters.

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

9.7.1. 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, such as 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 8.1.1).

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; however, without this knowledge, the author is obliged to either write text that they presume will be intercepted or 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 non-cryptographic MUA (i.e., PKCS #7 signed-data) or appear as an attachment that can cause confusion to a naive recipient using a non-cryptographic MUA (i.e., multipart/signed). If the composer 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). For example, how will the implementation indicate to the user when their own certificate is near expiry? 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. See [SMIME-SENDER-EXTENSIONS] for an example of how an MUA using a proxy protocol might indicate signing and encryption instructions to its proxy.

9.7.2. Dangers of an IMAP Proxy for Message Rendering

When receiving and rendering a message, the process of indicating the cryptographic status of a message to the user requires subtleties that are difficult to offer from a straightforward IMAP (or Post Office Protocol (POP) [RFC1939] 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 or inside the encryption?
- How should Errant Cryptographic Layers (see Section 4.5) be dealt with?
- What cryptographic protections do the header fields of the message have? (See [RFC9788].)
- 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 an unmodified header field through 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 an MUA to be confident in such a header field, it needs a guarantee from the proxy that any header field it produces will be safe. How does the MUA reliably negotiate 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 an inbound proxy handles certificate discovery in inbound messages (see Appendix A.3.1), it will also need to communicate the results of that discovery process to its corresponding outbound proxy for message composition (see Section 9.7.1).

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. [RFC9219] describes the transmission of an S/MIME signature verification status over JMAP, which is a subset of the cryptographic status information described here.

9.7.3. 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.

An MUA explicitly under the control of the end user with thoughtful integration can offer UI/UX and security guarantees that a simple proxy cannot provide. See also Appendix A.13 for suggestions of future work that might augment a proxy to make it safer.

9.8. Intervening MUAs Do Not Handle End-to-End Cryptographic Protections

Some MUAs will resend a message in identical form (or very similar form) to the way that they received it. For example, consider the following use cases:

- a mail expander or mailing list that receives a message and resends it to all subscribers (see also Appendix A.14 for more discussion of mailing lists)
- an individual user who reintroduces a message they received into the mail transport system (see Section 3.6.6 of [RFC5322])
- an automated email intake system that forwards a report to the mailboxes of responsible staffers

These MUAs intervene in message transport by receiving and then reinjecting messages into the mail transport system. In some cases, the original sender or final recipient of a message that has passed through such an MUA may be unaware of the intervention. (Note that an MUA that forwards a received message as a attachment (MIME subpart) of type `message/rfc822` or `message/global` or "inline" in the body of a message is *not* acting as an intervening MUA in this sense, because the forwarded message is encapsulated within a visible outer message that is clearly from the MUA itself.)

An intervening MUA should be aware of end-to-end cryptographic protections that might already exist on messages that they resend. In particular, it is unclear what the "end-to-end" properties are of a message that has been handled by an intervening MUA. For signed-only messages, if the intervening MUA makes any substantive modifications to the message as it passes it along, it may break the signature from the original sender. In many cases, breaking the original signature is the appropriate result, since the original message has been modified, and the original sender has no control over the modifications made by the intervening MUA. For signed-and-encrypted messages, if the intervening MUA is capable of decrypting the message, it must be careful when retransmitting the message. Will the new recipient be able to decrypt it? If not, will the message be useful to the recipient? If not, it may not make sense to resend the message.

Beyond the act of resending, an intervening MUA should not itself try to apply end-to-end cryptographic protections on a message that it is resending unless directed otherwise by some future specification. Additional layers of cryptographic protection added in an ad hoc way by an intervening MUA are more likely to confuse the recipient and will not be interpretable as end-to-end protections as they do not originate with the original sender of the message.

9.9. External Subresources in MIME Parts Break Cryptographic Protections

A MIME part with a content type that can refer to external resources (like `text/html`) may itself have some sort of end-to-end cryptographic protections. However, retrieving or rendering these external resources may violate the properties that users expect from cryptographic protection.

As a baseline, retrieving the external resource at the time a message is read can be used as a "web bug", leaking the activity and network location of the recipient to the server hosting the external resource. This privacy risk is present, of course, even for messages with no cryptographic protections but may be even more surprising to users who are shown some level of security indicator about a given message.

Other problems with external resources are more specifically bound to cryptographic protections.

For example, a signed email message with a `text/html` part that refers to an external image (i.e., via `<img src="https://example.com/img.png">`) may render differently if the hosting web server decides to serve different content at the source URL for the image. This effectively breaks the goals of integrity and authenticity that the user should be able to rely on for signed messages, unless the external subresource has strict integrity guarantees (e.g., via [SRI]).

Likewise, fetching an external subresource for a signed-and-encrypted message effectively breaks goals of privacy and confidentiality for the user.

This is loosely analogous to security indicator problems that arose for web browsers as described in [MIXED-CONTENT]. However, while fetching the external subresource over https is sufficient to avoid a "mixed content" warning from most browsers, it is insufficient for an MUA that wants to offer its users true end-to-end guarantees for email messages.

A conformant composing MUA that applies signing-only cryptographic protection to a new email message with an external subresource should take one of the following options:

- pre-fetch the external subresource and include it in the message itself,
- use a strong integrity mechanism like Subresource Integrity [SRI] to guarantee the content of the subresource (though this does not fix the "web bug" privacy risk described above), or
- prompt the composing user to remove the subresource from the message.

A conformant composing MUA that applies encryption to a new email message with an external resource cannot depend on Subresource Integrity to protect the privacy and confidentiality of the message, so it should either pre-fetch the external resource to include it in the message or prompt the composing user to remove it before sending.

A conformant rendering MUA that encounters a message with end-to-end cryptographic protections that contain a subresource **MUST** either refuse to retrieve and render the external subresource or decline to treat the message as having cryptographic protections. For example, it could indicate in the Cryptographic Summary that the message is Unprotected.

Note that when composing a message reply with quoted text from the original message, if the original message did contain an external resource, the composing MUA **SHOULD NOT** fetch the external resource solely to include it in the reply message, as doing so would trigger the "web bug" at reply composition time. Instead, the safest way to deal with quoted text that contains an external resource in an end-to-end encrypted reply is to strip any reference to the external resource during initial composition of the reply.

10. IANA Considerations

This document has no IANA actions.

11. Security Considerations

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

12. References

12.1. Normative References

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

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

[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>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[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>.

[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>.

[RFC9598] Melnikov, A., Chuang, W., and C. Bonnell, "Internationalized Email Addresses in X.509 Certificates", RFC 9598, DOI 10.17487/RFC9598, May 2024, <https://www.rfc-editor.org/info/rfc9598>.

[RFC9788] Gillmor, D. K., Hoeneisen, B., and A. Melnikov, "Header Protection for Cryptographically Protected Email", RFC 9788, DOI 10.17487/RFC9788, August 2025, <https://www.rfc-editor.org/info/rfc9788>.

12.2. Informative References

[AUTOCRYPT] Autocrypt Team, "Autocrypt - Convenient End-to-End Encryption for E-Mail", <https://autocrypt.org/>.

[CERT-BEST-PRACTICE] Woodhouse, D. and N. Mavrogiannopoulos, "Recommendations for applications using X.509 client certificates", Work in Progress, Internet-Draft, draft-woodhouse-cert-best-practice-01, 25 July 2023, <https://datatracker.ietf.org/doc/html/draft-woodhouse-cert-best-practice-01>.

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

[CLEARTEXT-COPY] Gillmor, D. K., "Encrypted E-mail with Cleartext Copies", Work in Progress, Internet-Draft, draft-dkg-mail-cleartext-copy-01, 21 February 2023, <https://datatracker.ietf.org/doc/html/draft-dkg-mail-cleartext-copy-01>.

[DROWN] Aviram, N., Schinzel, S., Somorovsky, J., Heninger, N., Dankel, M., Steube, J., Valenta, L., Adrian, D., Halderman, J. A., Dukhovni, V., Käsper, E., Cohney, S., Engels, S., Paar, C., and Y. Shavitt, "DROWN: Breaking TLS using SSLv2", Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), pp. 689-706, August 2016, <https://drownattack.com/drown-attack-paper.pdf>.

[EFAIL] Poddebniak, D., Dresen, C., Müller, J., Ising, F., Schinzel, S., Friedberger, S., Somorovsky, J., and J. Schwenk, "Efail: Breaking S/MIME and OpenPGP Email Encryption using Exfiltration Channels", Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), pp. 549-566, August 2018, <https://www.usenix.org/conference/usenixsecurity18/presentation/poddebniak>.

[IKE] Felsch, D., Grothe, M., Schwenk, J., Czubak, A., and M. Szymane, "The Dangers of Key Reuse: Practical Attacks on IPsec IKE", Proceedings of the 27th USENIX Security Symposium, pp. 567-583, August 2018, <https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-felsch.pdf>.

[MIXED-CONTENT] Stark, E., Ed., West, M., Ed., and C. Lopez, Ed., "Mixed Content", W3C Candidate Recommendation Draft, February 2023, <https://www.w3.org/TR/2023/CRD-mixed-content-20230223/>. Latest version available at <https://www.w3.org/TR/mixed-content/>.

[OPENPGP-FORWARDING] Wussler, A., "Automatic Forwarding for ECDH Curve25519 OpenPGP messages", Work in Progress, Internet-Draft, draft-wussler-openpgp-forwarding-00, 10 July 2023, <https://datatracker.ietf.org/doc/html/draft-wussler-openpgp-forwarding-00>.

[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", Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), pp. 4175-4192, August 2023, <https://www.usenix.org/conference/usenixsecurity23/presentation/ising>.

[PKCS11] Bong, D., Ed. and T. Cox, Ed., "PKCS #11 Specification Version 3.1", OASIS Standard, July 2023, <https://docs.oasis-open.org/pkcs11/pkcs11-spec/v3.1/os/pkcs11-spec-v3.1-os.html>.

[REPLY] Müller, J., Brinkmann, M., Poddebniak, D., Schinzel, S., and J. Schwenk, "Re: What's Up Johnny?: Covert Content Attacks on Email End-to-End Encryption", Applied Cryptography and Network Security (ACNS 2019), Lecture Notes in Computer Science, vol. 11464, pp. 24-42, DOI 10.1007/978-3-030-21568-2_2, April 2019, <https://doi.org/10.1007/978-3-030-21568-2_2>.

[RFC1939] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, DOI 10.17487/RFC1939, May 1996, <https://www.rfc-editor.org/info/rfc1939>.

[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>.

[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>.

[RFC3370] Housley, R., "Cryptographic Message Syntax (CMS) Algorithms", RFC 3370, DOI 10.17487/RFC3370, August 2002, <https://www.rfc-editor.org/info/rfc3370>.

[RFC4511] Sermersheim, J., Ed., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, DOI 10.17487/RFC4511, June 2006, <https://www.rfc-editor.org/info/rfc4511>.

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

[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>.

[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>.

[RFC6532] Yang, A., Steele, S., and N. Freed, "Internationalized Email Headers", RFC 6532, DOI 10.17487/RFC6532, February 2012, <https://www.rfc-editor.org/info/rfc6532>.

[RFC7292] Moriarty, K., Ed., Nystrom, M., Parkinson, S., Rusch, A., and M. Scott, "PKCS #12: Personal Information Exchange Syntax v1.1", RFC 7292, DOI 10.17487/RFC7292, July 2014, <https://www.rfc-editor.org/info/rfc7292>.

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

[RFC7929] Wouters, P., "DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP", RFC 7929, DOI 10.17487/RFC7929, August 2016, <https://www.rfc-editor.org/info/rfc7929>.

[RFC8162] Hoffman, P. and J. Schlyter, "Using Secure DNS to Associate Certificates with Domain Names for S/MIME", RFC 8162, DOI 10.17487/RFC8162, May 2017, <https://www.rfc-editor.org/info/rfc8162>.

[RFC8621] Jenkins, N. and C. Newman, "The JSON Meta Application Protocol (JMAP) for Mail", RFC 8621, DOI 10.17487/RFC8621, August 2019, <https://www.rfc-editor.org/info/rfc8621>.

[RFC8823] Melnikov, A., "Extensions to Automatic Certificate Management Environment for End-User S/MIME Certificates", RFC 8823, DOI 10.17487/RFC8823, April 2021, <https://www.rfc-editor.org/info/rfc8823>.

[RFC9051] Melnikov, A., Ed. and B. Leiba, Ed., "Internet Message Access Protocol (IMAP) - Version 4rev2", RFC 9051, DOI 10.17487/RFC9051, August 2021, <https://www.rfc-editor.org/info/rfc9051>.

[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>.

[RFC9219] Melnikov, A., "S/MIME Signature Verification Extension to the JSON Meta Application Protocol (JMAP)", RFC 9219, DOI 10.17487/RFC9219, April 2022, <https://www.rfc-editor.org/info/rfc9219>.

[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/info/rfc9580>.

[SMIME-SENDER-EXTENSIONS] Melnikov, A., "JMAP extension for S/MIME signing and encryption", Work in Progress, Internet-Draft, draft-ietf-jmap-smime-sender-extensions-04, 3 August 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-jmap-smime-sender-extensions-04>.

[SPOOFING] Müller, J., Brinkmann, M., Poddebniak, D., Böck, H., Schinzel, S., Somorovsky, J., and J. Schwenk, ""Johnny, you are fired!" - Spoofing OpenPGP and S/MIME Signatures in Emails", Proceedings of the 28th USENIX Security Symposium (USENIX Security 19), pp. 1011-1028, August 2019, <https://www.usenix.org/system/files/sec19-muller.pdf>.

[SRI] Akhawe, D., Braun, F., Marier, F., and J. Weinberger, "Subresource Integrity", W3C Candidate Recommendation, June 2016, <https://www.w3.org/TR/2016/REC-SRI-20160623/>. Latest version available at <https://www.w3.org/TR/SRI/>.

[WEBKEY-SERVICE] Koch, W., "OpenPGP Web Key Directory", Work in Progress, Internet-Draft, draft-koch-openpgp-webkey-service-20, 2 June 2025, <https://datatracker.ietf.org/doc/html/draft-koch-openpgp-webkey-service-20>.

Appendix A. Future Work

This document contains useful guidance for MUA implementers, but it cannot contain all possible guidance. Future revisions of this document may want to further explore the following topics, which are out of scope for this version.

A.1. Webmail Threat Model

The webmail threat model for end-to-end cryptographic protections is significantly more complex than the classic MUA model. For example, the web server hosting the webmail interface could be a potential adversary. If the user treats the web server as a trusted party, but the web server violates that trust, the end-to-end cryptographic protections do not hold.

A future version of this document could include more detailed discussion of an adversarial threat model for end-to-end cryptographic protections in a webmail context.

A.2. Test Vectors

A future version of this document (or a companion document) could contain examples of well-formed and malformed messages using cryptographic key material and certificates from [RFC9580] and [RFC9216].

It may also include example renderings of these messages.

A.3. Further Guidance on Peer Certificates

A.3.1. Certificate Discovery from Incoming Messages

As described in Section 8.2.3, an incoming email message may have one or more certificates embedded in it. This document currently acknowledges that a rendering MUA should assemble a cache of certificates for future use, but providing more detailed guidance for how to assemble and manage that cache is currently out of scope.

Existing recommendations like [AUTOCRYPT] provide some guidance for handling incoming certificates about peers but only in certain contexts. A future version of this document may describe in more detail how these incoming certificates should be handled.

A.3.2. Certificate Directories

Some MUAs may have the capability to look up peer certificates in a directory, for example, via the Lightweight Directory Access Protocol (LDAP) [RFC4511], Web Key Directory (WKD) [WEBKEY-SERVICE], or DNS (e.g., SMIMEA [RFC8162] or OPENPGPKEY [RFC7929] resource records).

A future version of this document may describe in more detail what sources an MUA should consider when searching for a peer's certificates and what to do with the certificates found by various methods.

A.3.3. Checking for Certificate Revocation

A future version of this document could discuss how/when to check for revocation of peer certificates or of the user's own certificate.

Such discussion should address privacy concerns: What information leaks to whom when checking peer certificate revocations?

A.3.4. Further Peer Certificate Selection

A future version of this document may describe more prescriptions for deciding whether a peer certificate is acceptable for encrypting a message. For example, if the SPKI is an Elliptic Curve (EC) public key and the keyUsage extension is absent, what should the encrypting MUA do?

A future version of this document might also provide guidance on what to do if multiple certificates are all acceptable for encrypting to a given recipient. For example, the composing MUA could select among them in some deterministic way; it could encrypt to all of them; or it could present them to the user to let the user select any or all of them.

A.3.5. Human-Readable Names in Peer Certificates, Header Fields, and Address Books

In header fields such as `From` that may contain a `display-name` as described in Section 3.4 of [RFC5322], a malicious composer (or interfering adversary) may populate the `display-name` part with a human-readable name that does not at all match the actual name of the participant. Section 8.1.1 describes some matching rules relating peer certificates to email addresses (the `addr-spec` part of these email header fields) but does not contemplate matching `display-names` or other similar user-visible data elements. Section 6.4 describes how signature validation should confirm a binding between the `addr-spec` and the certificate itself, but it also does not contemplate matching `display-names` or other similar user-visible data elements. Depending on how the rendering MUA renders the `display-name` in a message's header field, that unvalidated field may present a risk of user confusion, which could break the intended end-to-end assurances. Yet both X.509 and OpenPGP certificate formats offer ways to provide cryptographically certified (though possibly not unique) comparable human-readable names. Additionally, many MUAs also include an address book or comparable feature that can make substantive connections between user-relevant identity labels and email addresses.

A human-readable name like a `display-name` does not have the property of global uniqueness that an `addr-spec` does, so reasoning about human-readable names and rendering them to the user as an element in a system providing end-to-end cryptographic assurance requires additional deliberate analysis.

A future version of this document might offer strategies for associating human-readable names from certificates (and features like address books) to the rendering of header fields that include `display-name`. Such guidance should be paired with an analysis of specific usability and security risks associated with these human-readable fields, as well as a description of how the recommended guidance mitigates those risks.

A.4. Further Guidance on Local Certificates and Secret Keys

A.4.1. Cross-MUA Sharing of Local Certificates and Secret Keys

Many users today use more than one MUA to access the same mailbox (for example, one MUA on a mobile device and another MUA on a desktop computer).

A future version of this document might offer guidance on how multiple MUAs attached to the same mailbox can efficiently and securely share the user's own secret key material and certificates between each other. This guidance should include suggestions on how to maintain the user's keys (e.g., avoiding certificate expiration) and safe secret key transmission.

A.4.2. Use of Smart Cards or Other Portable Secret Key Mechanisms

Rather than having to transfer secret key material between clients, some users may prefer to rely on portable hardware-backed secret keys in the form of smart cards, USB tokens, or other comparable form factors. These secret keys sometimes require direct user interaction for each use, which can complicate the usability of any MUA that uses them to decrypt a large number of messages.

Guidance on the use of this kind of secret key management is beyond the scope of this document, but future revisions may bring them into scope.

A.4.3. Active Local Certificate Maintenance

Section 8.2.2 describes conditions where the MUA **SHOULD** warn the user that something is going wrong with their certificate.

A future version of this document might outline how an MUA could actively avoid these warning situations, for example, by automatically updating the certificate or prompting the user to take specific action.

A.5. Certification Authorities

A future document could offer guidance on how an MUA should select and manage root CAs.

For example:

- Should the MUA cache intermediate CAs?
- Should the MUA share such a cache with other PKI clients (e.g., web browsers)?
- What distinctions are there between a CA for S/MIME and a CA for the Web?

A.6. Indexing and Search of Encrypted Messages

A common use case for MUAs is the search of existing messages by keyword or topic. This is done most efficiently for large mailboxes by assembling an index of message content rather than by a linear scan of all message content.

When message contents and header fields are encrypted, search by index is complicated. If the cleartext is not indexed, then messages cannot be found by search. On the other hand, if the cleartext is indexed, then the index effectively contains the sensitive contents of the message and needs to be protected.

Detailed guidance on the trade-off here, including choices about remote search vs. local search, are beyond the scope of this document, but a future version of the document may bring them into scope.

A.7. Cached Signature Validation

Asymmetric signature validation can be computationally expensive, and the results can also potentially vary over time (e.g., if a signing certificate is discovered to be revoked). In some cases, the user may care about the signature validation that they saw when they first read or received the message, not only about the status of the signature verification at the current time.

So, for both performance reasons and historical perspective, it may be useful for an MUA to cache signature validation results in a way that they can be easily retrieved and compared. Documenting how and when to cache signature validation, as well as how to indicate it to the user, is beyond the scope of this document, but a future version of the document may bring these topics into scope.

A.8. Aggregate Cryptographic Status

This document limits itself to consideration of the cryptographic status of single messages as a baseline concept that can be clearly and simply communicated to the user. However, some users and some MUAs may find it useful to contemplate even higher-level views of cryptographic status, which are not considered directly here.

For example, a future version of the document may also consider how to indicate a simple cryptographic status of message threads (groups of explicitly related messages), conversations (groups of messages with shared sets of participants), peers, or other perspectives that an MUA can provide.

A.9. Expectations of Cryptographic Protection

As mentioned in Section 2.3, the types of security indicators displayed to the user may vary based on the expectations of the user for a given communication. At present, there is no widely shared method for the MUA to establish and maintain reasonable expectations about whether a specific rendered message should have cryptographic protections.

If such a standard is developed, a future version of this document should reference it and encourage the deployment of clearer and simpler security indicators.

A.10. Secure Deletion

One feature many users desire from a secure communications medium is the ability to reliably destroy a message such that it cannot be recovered even by a determined adversary. In other contexts, a similar desired property is called "forward secrecy". Doing this with standard email mechanisms such as S/MIME and PGP/MIME is challenging because of two interrelated factors:

- A copy of an email message may be retained by any of the mail transport agents that handle it during delivery.
- The secret key used to decrypt an encrypted email message is typically retained indefinitely.

This means that an adversary aiming to recover the cleartext contents of a deleted message can do so by getting access to a copy of the encrypted message and the long-term secret key material.

Some mitigation measures may be available to make it possible to delete some encrypted messages securely, but this document considers this use case out of scope. A future version of the document may elaborate on secure message deletion in more detail.

A.11. Interaction with Opportunistic Encryption

This document focuses on guidance for strong, user-comprehensible end-to-end cryptographic protections for email. Other approaches are possible, including various forms of opportunistic and transport encryption, which are out of scope for this document.

A future version of this document could describe the interaction between this guidance and more opportunistic forms of encryption, for example, some of the scenarios contemplated in [CLEARTEXT-COPY].

A.12. Split Attachments

As noted in Section 7.2, the standard form for encrypted email messages is a single Cryptographic Envelope. In a scenario where multiple user agents are drafting a single encrypted message over low-bandwidth links, this can create a poor user experience, as each MUA has to retrieve the full message, including attachments, to modify the draft. Similarly, when retrieving a message with a large attachment, the receiving MUA might want to only render the Main Body Part and will have a significant delay in doing so if required to process the full message before handling.

Future work might include an attempt to standardize a mechanism that eases this use case, potentially at the risk of additional metadata leakage about the message (e.g., the size and number of message parts). Any such work should explicitly try to minimize the risks and concerns described in Section 7.2.

A.13. Proxy Extensions

As noted in Section 9.7, a proxy-based implementation can be a tempting approach. But its naive form is likely to be insufficient to provide safe end-to-end encrypted email.

A future version of this document, or a separate but related document, could try to outline the specific additional information, state, and network API surface that would be needed to allow an MUA to be safely integrated with an encryption provider. Any such work should try to address the potential problems described in Section 9.7.

A.14. Mailing Lists

Mailing lists offer challenging complications to any notion of end-to-end cryptographic protections. By default, there is some sort of intervening MUA (see Section 9.8), but more than that, user expectations about cryptographic protections might differ from normal messages, at least insofar as they understand they are writing to a mailing list. A particular challenge to the

notion of end-to-end cryptographic security with mailing lists is that a subscriber to a mailing list often does not know who else is subscribed to the mailing list. Another challenge is that, for some mailing lists, some subscribers might not have a valid, non-expired certificate.

Encryption can interact with mailing lists in different ways, depending on the use case of the list. It's not clear that there are any useful motivations for sending encrypted mail to a publicly archived mailing list. But an unarchived mailing list might want to provide confidentiality between all recipients, even if the recipients don't know for certain who all the other participants are. Or a mailing list with private archives might well decide that two "hops" of encryption (between the composer and the mailing list, and the mailing list and all the subscribers) are useful confidentiality measures even though they are not "end-to-end" in the sense of the composer directly to all recipients.

Similarly, cryptographic signatures may play different roles in a mailing list, depending on the list's communication goals. The mailing list itself might want to verify that an incoming message is cryptographically signed by an authorized sender before redistribution to the list subscribers. It might also want to pass along the composer's signature in a way that the subscribers can all verify it. Alternately, the mailing list might want to sign each redistributed message itself and change the message so it appears to come from the list rather than the original composer.

Yet another design for a mailing list with end-to-end cryptographic protections might involve redistributing shared secret keys to all recipients or using some sort of proxied re-encryption scheme, similar to [OPENPGP-FORWARDING].

A future version of this document, or a separate but related document, might describe some of these trade-offs and provide guidance for safely meeting common requirements or use cases when combining end-to-end cryptographic protections with mailing lists.

Acknowledgements

The set of constructs and recommendations in this document are derived from discussions with many different implementers, including Bjarni Rúnar Einarsson, Daniel Huigens, David Bremner, Deb Cooley, Eliot Lear, Fabian Ising, Heiko Schaefer, Holger Krekel, Jameson Rollins, John Levine, Jonathan Hammell, juga, Patrick Brunschwig, Paul Kyzivat, Pete Resnick, Roman Danyliw, Santosh Chokhani, Stephen Farrell, Thore Göbel, and Vincent Breitmoser.

Authors' Addresses

Daniel Kahn Gillmor (EDITOR)
American Civil Liberties Union
125 Broad St.
New York, NY 10004
United States of America
Email: dkg@fifthhorseman.net

Alexey Melnikov (EDITOR)
Isode Ltd
14 Castle Mews
Hampton, Middlesex
TW12 2NP
United Kingdom
Email: alexey.melnikov@isode.com

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