

Why Johnny Should Not Delegate Email Encryption to Gateways

Andris Suter-Dörig
ETH Zurich
Zurich, Switzerland
sandr@ethz.ch

Matteo Scarlata
ETH Zurich
Zurich, Switzerland
matteo.scarlata@inf.ethz.ch

Kenneth G. Paterson
ETH Zurich
Zurich, Switzerland
kenny.paterson@inf.ethz.ch

Abstract

Email encryption based on S/MIME and, to a lesser extent, PGP is widely deployed in enterprise environments, often through centralised server-side *encryption gateways* that decrypt and sign messages on behalf of end users. Gateways can also be run locally to implement encryption for email clients that do not natively support it.

We systematically analyse the security of four such products: SEPPmail, CipherMail, Cisco Email Security Appliance, and the Proton Mail Bridge. We identify 29 attacks which allow an adversary to fully decrypt messages and confuse users on the authenticity of received emails. To this end, we exploit the legacy cryptographic primitives in S/MIME and the backwards compatibility features for PGP, adapt well-known attacks such as EFMAIL (Poddebniak et al., USENIX Security 2018) to this setting, and expose novel attack vectors such as the error signals produced by email infrastructure.

Our findings show that moving cryptographic operations to gateways strips important signals that email clients rely on. This re-enables attacks like EFMAIL, which were largely considered resolved, and introduces exploitable features in how the security status of emails is communicated. Such systems also expose sensitive decrypted plaintext to intermediate email infrastructure, which was not designed to handle it. We conclude that delegating cryptographic operations to a gateway undermines the end-to-end security model that PGP and S/MIME were designed to provide.

1 Introduction

Despite the rise of instant-messaging platforms, email remains the dominant medium for business communication. A broad array of regulatory frameworks require that sensitive messages be encrypted in transit [26–30, 68, 72]. Even in the absence of explicit regulation, many organisations adopt email encryption as a matter of policy, driven by contractual obligations or internal risk assessments. Deploying end-to-end encryption at scale, however, is notoriously difficult [76]. Key management, certificate distribution, and client compatibility impose operational burdens. In practice, many organisations therefore centralise cryptographic operations in a server-side *encryption gateway*: an appliance that sits on the mail path, transparently encrypting and signing outbound messages, and decrypting and verifying inbound ones on behalf of end users. Products such as SEPPmail, CipherMail and Cisco Email Security Appliance follow this model.

Beyond enterprise environments, gateways also appear as a way to interface encrypted email systems with clients that do not support them natively. Proton Mail is a prominent example: users can access their encrypted mail either through Proton’s own clients or through the Proton Bridge together with a conventional email

client. The Bridge is effectively a gateway, but one that runs locally on the user’s machine rather than on a server.

The appeal of these gateways is clear: they provide encryption transparently, without requiring end users to understand key management, certificate validation, or the underlying email encryption standards. From an organisational perspective, a gateway reduces the deployment problem from “train every user” to “configure one appliance”. SEPPmail alone counts thousands of installations across banks, insurance companies, government offices and healthcare providers in Europe, particularly in the German-speaking (DACH) region [62]. CipherMail is an open-source alternative in the DACH region and Benelux—while we were not able to confirm precise numbers, they claim that their customers include banks, law firms, pension funds, government agencies, and healthcare providers [10]. Cisco integrates its Email Security Appliance service into a broader email security platform deployed at enterprise scale for hundreds of large business customers, mostly in the United States [25]. Proton Mail counts over 100 million users, but does not release data on how many of these use Proton Bridge [50]. Together, these four products span the gateway design space: they vary by topology (server-side vs. local), crypto stack (BouncyCastle, OpenSSL, GopenPGP), protocol (S/MIME+PGP, S/MIME-only, PGP-only), and market (European/DACH, large US enterprises, consumer).

However, these gateways also introduce a fundamental tension: they perform exactly the operations that end-to-end encryption is designed to keep away from intermediaries. This not only creates new attack surfaces that do not exist in a true end-to-end model, but also introduces a single point of failure where a vulnerability may affect the security of all users in an organisation simultaneously rather than on an individual basis. In this paper we show that this architectural choice re-opens attack vectors that client-side mitigations had closed, and introduces entirely new ones. Concretely, we make the following contributions:

- We demonstrate that decryption on a gateway strips the signals that downstream clients rely on to activate EFMAIL mitigations, effectively reviving attacks that were previously considered resolved.
- We show that gateways behave in unexpected ways when handling messages with mixed content, where encrypted, plaintext, and signed fragments are interleaved. This yields attacks based on inline-PGP splicing and partial signatures, and a novel MIME boundary-injection vector that turns the gateway’s decrypt-and-reserialize step into a plaintext-exfiltration channel.
- We identify, for the first time, bounce-handling behaviours that yield trivial or oracle-based plaintext recovery, including full-message bounces, header-decryption via S/MIME

ciphertext malleability, and padding oracles. Padding oracle attacks requiring large numbers of queries—impractical in the end-to-end setting because they would depend on repeated user interaction—become possible because the gateway performs cryptographic operations automatically. In classic PGP and S/MIME deployments, clients usually decrypt an email only when it is viewed by the user. Gateways, on the other hand, decrypt and process emails automatically as soon as they are received.

- We show that the gateways do not have reliable mechanisms to signal the encryption and signature status of incoming emails to end users. Many products rely on custom tags in the subject line; we show that these can be manipulated through Unicode lookalikes, bounded-sanitisation bypasses, and tag-hiding techniques.
- We systematically evaluate four gateway encryption products against these four classes of attacks: revived EFAIL exfiltration, mixed-content confusion, bounce-based decryption oracles, and spoofing user-signalling. We report a total of 29 attacks across all the products: 11 against SEPPmail, 12 against CipherMail, 4 against Cisco, and 2 against the Proton Mail Bridge.

Methodology. We deliberately limit our scope to cryptographic attack vectors, but expect that many other types of attacks may be possible. Our analysis follows the “attack” branch of the Cryptography in the Wild methodology [1], applied to email encryption gateways. We select four products that span the main gateway deployment variants in practice and, using the relevant standards, vendor documentation, and source code where available (and reverse-engineering where it is not), model how each one receives, decrypts, verifies, rewrites, and forwards mail. We then look for weaknesses created by the interaction between cryptography and ordinary mail-processing features such as HTML rendering, MIME assembly, bounce generation, and subject-line tagging. Each attack reported in this paper was validated with a proof-of-concept against an isolated self-hosted or trial deployment. The 29 attacks reported here should be read as lower bounds rather than complete vulnerability counts. For example, our broader six-month analysis of SEPPmail [67] alone resulted in 56 findings, of which this paper discusses only the subset most relevant to our cryptographic focus. For all the affected vendors, we have undergone a coordinated disclosure process. Exact product versions can be found in Appendix C. We discuss the disclosure process at greater length in Appendix E.

Limitations. Beyond our focus on cryptographic attack vectors, our selection of gateway products was limited by software availability. We also do not measure market share: gateways are often deployed within company networks, we can only rely on vendor- and third-party-reported figures to estimate their prevalence. Some of our attacks rely on non-default configurations, which we note in the per-attack discussions. Others target the mechanisms for signalling the security status of emails to users; for these, we did not conduct a user study to measure whether end users actually notice forged or suppressed tags.

Roadmap. The remainder of this paper is organised as follows. Section 1.1 discusses related work. Section 2 provides background

on email structure, encryption standards, and introduces the threat model and the products we analyse. Section 3 presents our attacks in four subsections. We explore mitigations in Section 4, and discuss wider implications of our work in Section 5. Section 6 concludes.

1.1 Related Work

There is a long history of cryptographic analysis of PGP and S/MIME standards and implementations [8, 23, 35, 37–39, 42, 43, 47, 49]. The most directly related line of work is EFAIL [49], which demonstrated that both S/MIME and OpenPGP were, in 2018, vulnerable to chosen-ciphertext exfiltration attacks exploiting the malleability of CBC-mode encryption and the lack of authenticated encryption in legacy PGP. Schwenk et al. [59] analysed client-side mitigations and proposed a classification of EFAIL defence strategies. Our work shows that all such mitigations are rendered ineffective when a gateway decrypts the message before the client ever sees it, because the signal that triggers the mitigation is removed.

In Germany, healthcare providers exchange sensitive data through KIM [32] (*Kommunikation im Medizinwesen*), a large-scale S/MIME deployment. Saatjohann et al. [56] analysed the KIM specifications and real-world client stacks, reporting implementation and certification gaps despite a strong cryptographic baseline. Saatjohann [55] presented issues in KIM 1.5. Groves and Wachter [33] presented the `gpg.fail` catalogue of OpenPGP implementation flaws, some of which relate to how the presentation of cryptographic signals to users can be manipulated.

Ising et al. [35] demonstrated format oracles in email end-to-end encryption, showing that differences in how mail clients handle malformed encrypted messages can leak information about the plaintext structure. Their work is complementary to ours: while format oracles exploit client-side parsing differences, our bounce and padding oracles exploit gateway-side processing behaviour. Öndarö et al. [48] performed a large-scale measurement of S/MIME certificates deployed on the Internet, providing empirical evidence for the scale and characteristics of S/MIME deployment. Jager, Paterson, and Somorovsky [36] showed how backwards-compatibility features in cryptographic protocols enable downgrade attacks in the W3C XML encryption standards. We show that similar attacks apply to S/MIME. Strenzke [65] identified an existential forgery vulnerability in CMS and PKCS#7 signatures.

A long line of usability studies—beginning with Whitten and Tygar’s seminal “Why Johnny Can’t Encrypt” [76] and surveyed systematically by Clark et al. [21]—has repeatedly shown that end users struggle with key management, trust decisions, and interpreting security indicators. Related work by Müller et al. [45] showed that OpenPGP and S/MIME signatures could be spoofed in many email clients. In separate work, Müller et al. [46] demonstrated covert content attacks on email end-to-end encryption. Email gateways were partly motivated by the desire to remove usability burdens, yet, as we show, doing so introduces a different class of vulnerabilities.

2 Background

2.1 Email Fundamentals

MIME structure. Email messages are generally encoded using the *Multipurpose Internet Mail Extensions* (MIME) standard defined in RFC 2045 [31]. A MIME message consists of a tree structure of

MIME entities. In turn, each MIME entity has a set of header fields consisting of key-value pairs followed by an empty line and then a body. The type of the body is specified in the Content-Type header field. If the content type is any of the `multipart/*` types, the body in turn consists of a sequence of further MIME entities separated by the specified boundary string as shown in Listing 1. Such nested MIME structures allow for images or attachments to be embedded in the email or, as in the example, to provide alternative versions of the same message in different formats.

In order for this to work, the boundary string must not appear in any of the enclosed parts. Several strategies exist to ensure this: (1) the boundary string is chosen in such a manner that the escaped body cannot contain it; (2) the body is scanned for the boundary string: if it is found, a new boundary string is chosen; (3) the boundary string is sampled at random from a large pool of possible boundary strings: if done correctly, the probability of the body containing the boundary string is negligible.

Listing 1: MIME email structure.

```
Subject: Example Message
From: sender@example.com
To: recipient@example.com
Date: Thu, 9 Apr 2026 20:45:39 -0400
Content-Type: multipart/alternative;
    boundary="bound123"

--bound123
Content-Type: text/plain

This is just an example message
--bound123
Content-Type: text/html

<p>This is just an example message</p>
--bound123--
```

Email forwarding. In practice a single email may traverse several servers, known as *Mail Transfer Agents* (MTAs), before being delivered to the recipient’s *Mail User Agent* (MUA). This allows for setups such as the one depicted in Figure 1, where the email is processed by various servers with distinct responsibilities. By default, when one of the MTAs is unable to process or deliver an email, it will generate a *Delivery Status Notification* (DSN) [44] and send it to the sender of the original email. Depending on a number of factors, the DSN may contain either the entire original email or only the headers. This can help identify the root cause of the delivery failure and allow the issue to be resolved.

DKIM. *DomainKeys Identified Mail* (DKIM) [22] adds a signature to an email to allow the recipient to check that an email was authorised by the sending domain and handled by an appropriate outbound MTA. The signature is added by the sender’s MTA and covers the body of the email as well as selected headers. The recipient MTA can then look up the sender domain’s public key using DNS and verify the signature to ensure that covered parts of the email have not been modified in transit.

2.2 Email Encryption

The two dominant standards for email encryption are OpenPGP and S/MIME. From a user’s perspective, the primary difference between the two is how they handle key distribution. S/MIME

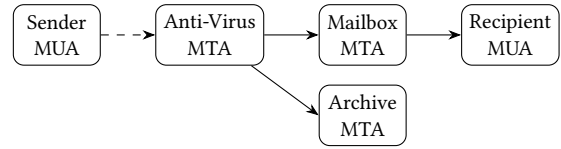


Figure 1: Example email-forwarding chain.

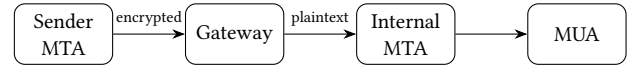


Figure 2: Gateway deployment with gateway-side decryption.

relies on the same *Public Key Infrastructure* (PKI) as TLS, using X.509 certificates [5] issued by a *Certificate Authority* (CA). PGP, by contrast, adopts a decentralised model in which users distribute their own public keys and establish authenticity through out-of-band direct verification or the *Web of Trust* [70].

S/MIME. The S/MIME specification [58] provides structures to encrypt or sign MIME entities and then re-encode the result into a new MIME message. To this end, it defines how to serialise the cryptographic primitives using the *Cryptographic Message Syntax* (CMS) format [34] into *wrapper* emails. However, the CMS format is very generic and applications need to take care when instantiating it to prevent vulnerabilities. For example, while S/MIME version 4.0 has added the option of using authenticated encryption, it still requires support for unauthenticated encryption modes, which allows for several backwards compatibility attacks discussed in Section 3.3.

OpenPGP. Unlike S/MIME, OpenPGP [77] (also known simply as PGP) does not target emails exclusively, but provides a more generic framework for encrypting and signing arbitrary messages and documents. The outputs of such PGP operations are called PGP messages. There are two common ways to use OpenPGP for email encryption. First, the OpenPGP specification supports encoding PGP messages in a text format. Such text can then simply be sent as the body or attachment of an email. This approach is known as *inline-PGP*. Alternatively, RFC 3156 [54] describes how to use PGP to provide protection for MIME entities in a similar way to S/MIME. We will refer to the email carrying such protected MIME entities as *wrappers*. This approach is commonly referred to as *PGP/MIME*. An important difference between S/MIME and PGP is that the current version of PGP largely disallows the use of non-authenticated encryption.

2.3 Target Products

We evaluate four products that span the spectrum of gateway encryption architectures. The first three share a common deployment pattern (see Figure 2): an encryption gateway is part of the organisation’s internal mail infrastructure, sitting between external MTAs on the public internet and the internal MTA responsible for managing user mailboxes. Inbound messages encrypted with S/MIME or PGP arrive from the external MTA; the gateway decrypts them and verifies the signature on the message. It then forwards the resulting plaintext to the internal MTA, which delivers it to the user’s MUA. On the outbound path the gateway may encrypt and sign plaintext messages before they leave the organisation.

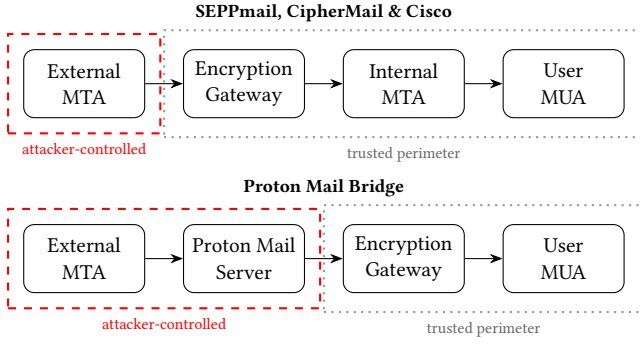


Figure 3: Trust boundary.

SEPPmail. SEPPmail [61] is a commercial MTA-relay appliance that implements the architecture described above. It supports S/MIME, PGP/MIME, and inline-PGP for both encryption and signatures. When a message is successfully decrypted or its signature is verified, SEPPmail adds a configurable tag to the Subject header, so that the end user can check the message’s cryptographic status. For example, by default, [signed OK] is appended to properly signed messages.

CipherMail. CipherMail [14] is an open-source email encryption gateway (with a commercial edition available) that also implements the architecture described above. It supports S/MIME, PGP/MIME, and inline-PGP. CipherMail provides similar subject-line tagging to SEPPmail. However, this feature is disabled by default. Instead, users would have to manually check custom headers in the email to see whether emails are properly secured.

Cisco Email Security Appliance. Cisco’s *Email Security Appliance* (ESA) [18] is also a commercial MTA-relay appliance. It supports S/MIME but not PGP. It does not provide dedicated means to signal the cryptographic status of incoming emails to end users. However, administrators can manually implement a tag policy using generic header manipulation tools and filters.

Proton Mail Bridge. Proton Mail [53] is an end-to-end encrypted mailbox provider. Its *Bridge* [51] component exposes a local email server interface to conventional mail clients and encrypts or decrypts traffic on that leg as messages move between the MUA and Proton Mail’s servers. The Bridge is thus an encryption/decryption gateway, loosely analogous to the gateway MTAs discussed above, even though it is supposed to run on the user’s machine rather than as a relay in the organisation’s email path.

Proton Mail exclusively supports PGP, but does not internally employ the MIME message format. The Proton Mail servers preprocess incoming email into a custom message structure consisting of multiple fragments, roughly analogous to MIME parts, each of which consists of a PGP encrypted body and unprotected headers. The gateway fetches messages from the servers using TLS, decrypts them and packages them up in an email format for the user’s MUA.

The Proton Mail Bridge does not implement the subject-line user-signalling described for SEPPmail and CipherMail. Bridge-delivered emails therefore have no dedicated, client-verifiable indicator of cryptographic status.

2.4 Threat Model

Since the requirements for individual attacks differ, we consider attackers with various combinations of the following capabilities. Each capability is given a short code (e.g., CTX) that we reuse in the attack sections and in Table 1 to indicate which capabilities each attack requires.

- CTX** Access to a captured ciphertext that the attacker wants to decrypt, where the intended recipient of that ciphertext is a user of the targeted gateway.
- SIG** Access to a captured signature from the user the attacker wishes to impersonate. Notably, this can also be a signature on an email sent to the attacker intentionally.
- SPF** The ability to bypass DKIM/SPF sender-authentication checks. This can be achieved, for instance, by compromising the right MTA or because the checks are not strictly enforced.¹
- DSN** Access to the DSNs sent by either the encryption gateway or an internal MTA. Notably, this can include DSNs sent to an attacker-specified email address.
- MUA** For some attacks, the victim’s MUA must be configured to load remote content.

We do *not* assume that the adversary has compromised the gateway itself, the internal mail infrastructure behind it, or the end user’s device. Control of an on-path MTA is one concrete way to obtain most of these capabilities. An adversary controlling an MTA on the external (marked in red in Figure 3) route can capture ciphertexts and signatures and can intercept, read, replay, and inject messages. It also can read DSNs and pass SPF checks; if the adversary controls the sender’s outbound MTA, it can then trivially pass DKIM checks. If an MTA-to-MTA link is unprotected,² we can additionally relax these assumptions: any on-path network adversary can do the above without control of MTAs, with the exception of bypassing DKIM.

The Proton Mail Bridge runs locally and delivers mail directly to the MUA without an internal MTA. For Proton Mail, we therefore assume an adversary that controls the Proton Mail server, which is sufficient to obtain the capabilities our attacks require. Unlike the gateways, the Proton Mail server lies outside the trusted perimeter, for two reasons. First, Proton advertises end-to-end security, so its servers are untrusted by design. Second, this placement most directly mirrors the threat model we assume for SEPPmail, CipherMail and Cisco, as illustrated by Figure 3.

3 Attacks on Email Encryption

We evaluate the four selected email encryption gateways against four classes of attacks. We begin with attacks that revive the EFAIL exfiltration techniques in Section 3.1, then examine how mixed encrypted and plaintext content is handled in Section 3.2. Section 3.3 discusses how bounce-message behaviour can be exploited to construct decryption oracles. Finally, Section 3.4 targets the subject-line tags that gateways use to communicate encryption and signature status to end users. In each subsection a summary table indicates

¹In a 2022 study Wang et al. report just 28% DKIM adoption among Alexa Top 1M domains [74].

²This is often the case: Bleichschmidt and Stock show that 30% of TLS certificates for MTA-to-MTA links are invalid [3].

Table 1: Consolidated results across all attack categories. SEPP for SEPPmail, CM for CipherMail, ESA for Cisco ESA and PrBr for Proton Bridge. In the *Non-default settings / Notes* column, upright text lists the configuration that must be changed from its default to enable the attack, while *grey italics* denote general remarks such as available mitigations or caveats. Every ✓ marks an attack for which a PoC was implemented. The *Outcome* column lists the high level consequences of the attack and, where relevant, the plaintext recovery rate in bytes per probe emails. The *Adv Cap* column lists adversarial capabilities (see § 2.4).

	Attack Class	Prior Art	Target	Non-default settings / Notes	PoC	Outcome (Rate, B/#emls)	Adv Cap
Reviving EFMAIL (§ 3.1)	S/MIME Gadget	[49]	SEPP	–	✓		
			CM	– / <i>Partial mitigation: reject decrypted S/MIME emails that contain invalid bytes.</i>	✓	ptxt recovery	CTX, MUA
			ESA	–	✓		
	PGP Gadget	[49]	CM	– / <i>Mitigation: require authenticated encryption for PGP.</i>	✓	ptxt recovery ptxt modification	CTX, MUA
	PGP Direct Exfiltration	[49]	CM	Enable inline-PGP processing. / <i>Partial mitigation: add mixed content warnings (bypassable).</i>	✓	ptxt recovery	CTX, MUA
			PrBr	– / <i>Malicious server</i>	✓		
Mixed Content (§ 3.2)	Mixed Ciphertext	[49]	SEPP	–	✓		
			CM	Enable inline-PGP processing. / <i>Partial mitigations: add mixed content warnings (bypassable); disable Scan HTML for ptxt recovery.</i>	✓	ptxt modification ptxt recovery	CTX, MUA
	Partial Signature	–	SEPP	–	✓		
			CM	Enable inline-PGP processing. / <i>Partial mitigation: add mixed content warnings (bypassable)</i>	✓	sig. forgery	SIG, SPF
	MIME boundary	–	SEPP	–	✓		
			CM	–	✓	ptxt recovery	CTX, MUA
			PrBr	– / <i>Malicious server</i>	✓		
Exploiting Bounces (§ 3.3)	Full Message	–	SEPP	–	✓		
			CM	– / <i>Particularly affected: generates full message bounces itself.</i>	✓	ptxt recovery (∞/1)	CTX, DSN
			ESA	– / <i>Particularly affected: generates full message bounces itself.</i>	✓	(∞/1)	
	Header Decryption	–	SEPP	–	✓		
			CM	– / <i>Lack of CFB-mode support prevents the GCM-to-CFB downgrade variant.</i>	✓	ptxt recovery (16 B/1)	CTX, DSN
			ESA	–	✓	(16 B/1)	
	Padding Oracle	[36, 73]	SEPP	– / <i>In version ≥ 15.0.3, multiple probes can be sent in a single email, massively reducing the number of emails that need to be sent.</i>	✓	(16 B/17)	
			CM	– / <i>If enabled, the valid character check can be used to significantly improve the attack’s efficiency.</i>	✓	ptxt recovery (16 B/143)	CTX, DSN
			ESA	–	✓	(16 B/2058)	
User Signalling (§ 3.4)	Unicode spoofing	[45, 56]	SEPP	–	✓		
			CM	Enable tagging signed emails.	✓	sig. forgery	SPF
	Bounded sanitisation	[45, 56]	SEPP	–	✓		
			CM	Enable tagging signed emails.	✓	sig. forgery	SPF
	Tag hiding	[45, 56]	SEPP	–	✓		
			CM	Enable tagging signed emails.	✓	ptxt modification sig. forgery	SPF

which products are affected (✓), or unaffected (✗), for each of the discussed attacks. Every ✓ reported in these tables is supported by a proof-of-concept implementation; we discuss the concrete performance of our PoCs in Appendix A. Attacks may also be not applicable (∼) to the product because it does not support a relevant feature. Table 1 provides a consolidated view of all the attacks.

3.1 Reviving EFAIL

Table 2: EFAIL results. All the attacks exfiltrate plaintext.

	S/MIME Gadget	PGP Gadget	PGP Direct
SEPPmail	✓	✗	✓
CipherMail	✓*	✓*	✓*
Cisco	✓	∼	∼
Proton Mail	∼	✗	✓

* Partial mitigations are available.

The EFAIL attacks [49] allow an attacker who has intercepted an encrypted email to recover its plaintext after inducing a single decryption by the victim’s client. All three variants discussed below require a captured ciphertext and a victim MUA that loads remote content (CTX, MUA). The key insight of [49] is that emails are not inert containers: MUAs render HTML, follow remote resource references, and splice together content across multiple MIME parts. Encrypted emails inherit exactly this rich feature set. An attacker who can arrange for the decrypted plaintext to land inside an HTML construct that triggers an outbound request—most commonly the `src` attribute of an `<img>` tag—causes the MUA to send the plaintext to an attacker-controlled server as part of a normal HTTP fetch. The authors of [49] present two complementary variants that differ in how the decrypted plaintext is placed into the exfiltration URL. Each applies to both S/MIME and OpenPGP.

Direct exfiltration (see Figure 4, top) targets MIME composition rather than the cryptographic layer. The attacker rewraps the intercepted ciphertext in a crafted `multipart/mixed` message with three sub-parts.

- (1) The first part opens an HTML `<img>` tag with an unterminated `src="https://attacker.com/... attribute`.
- (2) The second part contains the intercepted, encrypted PGP or S/MIME ciphertext.
- (3) The third closes the `src` attribute and the `<img>` tag with `>`.

If the MUA, after decrypting the middle part, renders all three parts within a single HTML context, the decrypted plaintext is spliced between the attacker’s fragments, yielding a well-formed `<img>` whose source points to an attacker-controlled server. Since the path of the URL contains the plaintext that the MUA decrypted, plaintext is sent to the attacker when the MUA loads the image. No ciphertext manipulation is required: the attack exploits only the fact that, at the time of disclosure, several MUAs combined sibling MIME parts into a single HTML rendering context even when one of those parts had just been decrypted from an S/MIME or PGP wrapper.

The *gadget* variant (see Figure 4, bottom) instead manipulates the ciphertext itself. At the time, both standards admitted unauthenticated symmetric modes: for example, CBC in S/MIME and CFB in OpenPGP. In either mode, an attacker who knows a plaintext

block P_i can XOR a chosen differential $P_i \oplus G$ into the appropriate ciphertext block so that the block at position i decrypts to an attacker-chosen *gadget* G at the cost of one adjacent block’s decryption being garbled. By splicing such gadgets before and after a target region of ciphertext, the attacker constructs, purely through ciphertext manipulation, the same `` shape as in the direct variant—modulo block alignment and the garbling introduced by each XOR. HTML parsers tolerate garbled bytes inside a quoted attribute value, so the corrupted blocks between gadgets do not prevent the MUA from requesting the image and, thereby, sending the secret to the attacker.

Reviving EFAIL. In response to the EFAIL disclosure, major MUAs deployed countermeasures that we revisit in Section 5. Thunderbird, for example, stopped rendering multiple MIME parts in a single HTML context, which defeats the direct exfiltration variant. Other MUAs apply their defences more selectively: Apple Mail, for instance, replaces a decrypted MIME part with a single synthetic HTML element that does not expose any of the plaintext when included in a URL. Some of the mitigations, such as the one used by Apple, require that the client recognises that (part of) the message came out of an S/MIME or PGP wrapper. An encryption gateway eliminates exactly this signal by transparently decrypting the message and handing the MUA a plain text/html body with the S/MIME or PGP encapsulation already stripped. Because of this, the gateways we study in the remainder of this section revive both EFAIL variants against clients that would otherwise be protected.

Results. Not every EFAIL variant applies to every product. For example, none of the products we tested are vulnerable to the direct exfiltration variant for S/MIME since, if they support S/MIME in the first place, their S/MIME implementations do not support the required MIME structure. In most cases, they inherit this lack of support for more complex MIME structures from the limited MIME parser provided by OpenSSL [69] since they rely on it for their S/MIME implementation.

Proton Mail Bridge does not support S/MIME, but the S/MIME gadget attack affects all the other products. CipherMail offers an explicit EFAIL countermeasure: optional post-decryption checks that flag bytes outside the S/MIME 7-bit envelope expected of message bodies [9]. If one is found, CipherMail can, for example, abort the decryption and deliver the still encrypted message instead. In the CipherMail version we tested, this mitigation is disabled by default and only available for S/MIME.

Cisco ESA does not support PGP, so neither PGP EFAIL variant applies to it. The PGP gadget attack is unique among the issues we discuss: SEPPmail and Proton Mail support PGP decryption, yet neither is vulnerable. Both reject the unauthenticated ciphertexts required by the attack, consistent with the current OpenPGP standard [77]. CipherMail’s default configuration violates the RFC, leaving it vulnerable.

While plain PGP Direct exfiltration works against SEPPmail, the exploitability depends on deployment details for CipherMail instances. Its `[mixed content]` warning can indicate suspicious composition, but this control is both bypassable (see Section 3.2) and disabled by default. In addition, the attack path requires incoming inline-PGP processing to be enabled.

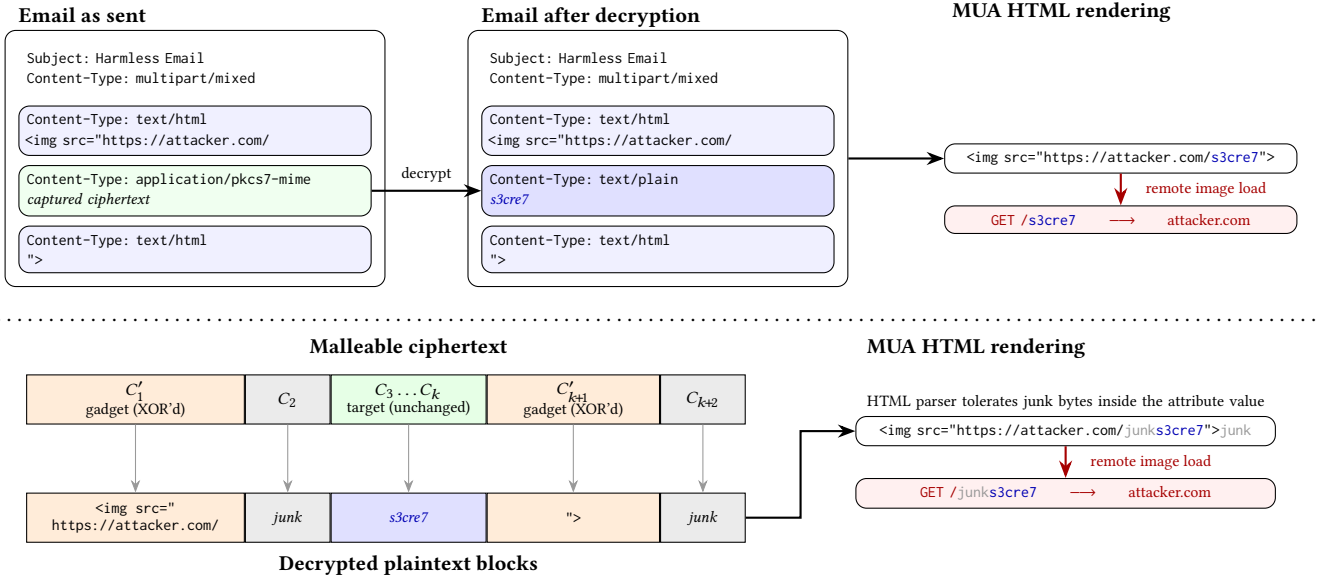


Figure 4: EFMAIL mail structure for direct exfiltration (top half) and gadget (bottom half) variants.

The Proton Mail Bridge has some mitigations against the direct exfiltration variant of EFMAIL. For each MIME part, the server uses some proprietary heuristics to either

- (1) mark it as the main message body,
- (2) mark it as an attachment with inline disposition (e.g. images displayed in the main body but not listed as an attached file),
- (3) mark it as an attachment with attachment disposition (e.g. actual attached files that are not displayed in the main body),
- (4) or drop it from the email entirely.

It differentiates between the second and third case by setting the Disposition field for the respective Attachment object. Since attachments with an attachment disposition are not displayed in the email body, they cannot be used for the direct exfiltration attack. Therefore, the Proton Mail Bridge only adds additional MIME structure to prevent the attack for parts with inline disposition. However, to determine this, it checks whether the disposition is exactly the string inline. Many MUAs render parts with other disposition values in the message body, creating an interpretation mismatch that allows a malicious server to bypass the Bridge’s mitigation and mount a PGP direct-exfiltration EFMAIL attack.

3.2 Mixed Content

In this subsection, we study *mixed content*: single delivered messages in which multiple ciphertext, plaintext, and signed fragments coexist. In such cases, a gateway’s pipeline of decryption, verification, and outgoing MIME assembly can obscure which bytes originated from the legitimate sender and which were supplied by an adversary. Table 3 shows our results for three issues along that pathway: ciphertext splicing, partial signing, and MIME boundary handling. We discuss each issue in turn.

3.2.1 Mixed Ciphertext. When a gateway decrypts inline-PGP blocks embedded alongside unencrypted text and replaces them

Table 3: Mixed content results. Mixed Ciphertext and MIME Boundary Injection attacks lead to plaintext exfiltration; Partial Signature attacks invalidate the integrity and authenticity guarantees of received emails.

	Mixed Ciphertext	Partial Signature	MIME boundary Injection
SEPPmail	✓	✓	✓
CipherMail	✓*	✓*	✓
Cisco	~	~	~
Proton Mail	~	~	✓

* Mitigations are available.

with the resulting plaintext, this obscures which bytes of the resulting email were or were not originally encrypted. The user receives a single message in which attacker-controlled plaintext and legitimately encrypted content are merged without distinction. This allows an attacker to prepend or append data to the plaintext of an intercepted PGP-encrypted block. Like the EFMAIL variants, this attack requires a captured ciphertext and a victim MUA that loads remote content (CTX, MUA).

Consequently, the adversary can alter how the message is interpreted in ways that properly used authenticated encryption should prevent. This is particularly problematic for HTML emails where embedding a PGP ciphertext in an attacker-controlled context allows building the same type of exfiltration channel used by the direct exfiltration variant of the EFMAIL attacks (see Section 3.1). When the MUA renders the HTML shown in Figure 5, it will send the contents of the decrypted plaintext to `attacker.com`.

Results. Neither Cisco’s ESA nor the Proton Mail Bridge implements inline-PGP, so this form of ciphertext-mixing attack does

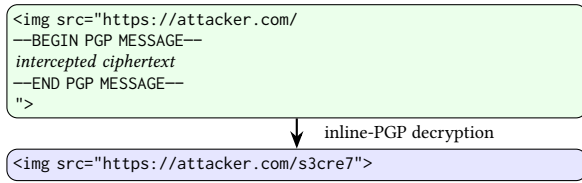


Figure 5: Mixed content decryption

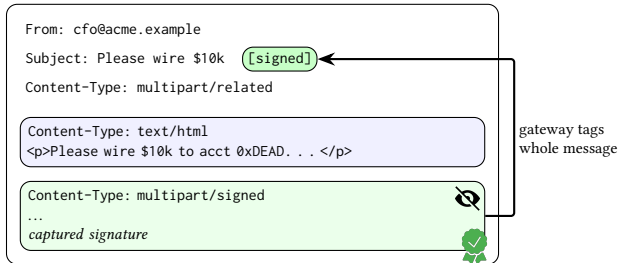


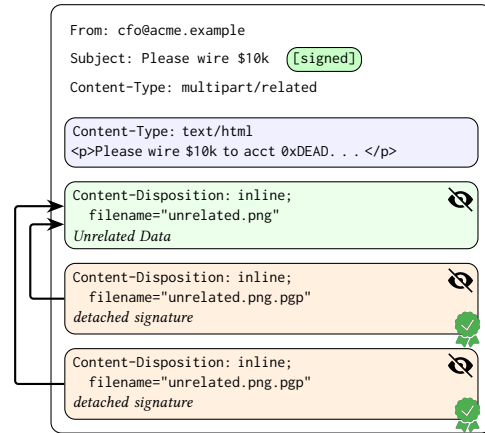
Figure 6: Partial signature attack.

not apply. SEPPmail is vulnerable in its default configuration. CipherMail does support inline-PGP, but ships with it disabled by default for performance reasons [15]. Furthermore, on CipherMail the automatic exfiltration attack described above requires the *Scan HTML* [16] setting to be **disabled**. This is because the effect of this somewhat misnamed setting is to convert any HTML that contains inline-PGP ciphertext into plain text.

CipherMail can also be configured to add a `[mixed content]` tag to an email’s subject line when it detects mixed content. A well-informed user could then use this tag to realise they must treat the email with caution. However, CipherMail only considers a MIME part to be mixed content when it combines an inline-PGP ciphertext with (non-whitespace) plaintext. Since an attacker can simply encrypt any such plaintext with the victim’s public key themselves, this check can easily be bypassed.

3.2.2 Partial signatures. Here we exploit the fact that gateways may tag an entire message as signed even when only a subset of its MIME parts carry a valid signature. An attacker constructs a multipart message in which one part carries a legitimate signature (e.g., copied from another message sent by the party they wish to impersonate), while the remaining parts contain attacker-chosen content. The gateway verifies the signature on the signed part, finds it valid, and tags the entire message as signed. Notably, the MIME structure can be chosen so that the user’s MUA never displays the signed part—for example, an inline image which is never actually referenced from any HTML (see Figure 6). This allows an attacker to forge an effectively arbitrary signed message from any sender from which they have obtained a signed message. The attack requires a captured signature from the impersonated sender and the ability to bypass DKIM/SPF checks (SIG, SPF).

Results. Neither Cisco’s ESA nor the Proton Mail Bridge surfaces per-message signature status to the recipient, so the partial-signature attack has no observable effect against either product. This means that the attack does not apply, but also that the end user has no

Figure 7: Partial signatures in CipherMail: with two unsigned parts and two detached signatures, the `[mixed content]` tag is omitted.

indication of signature correctness. This is the same pattern we revisit in Section 3.4: the absence of a tag denies the attacker a target but also leaves the recipient without any indication of the message’s cryptographic status.

SEPPmail is affected by a straightforward version of the attack. Once any well-formed signature in the message verifies, the gateway appends its `[signed OK]` tag to the subject line without recording which MIME parts the signature actually covered. An attacker can therefore attach an arbitrary unsigned part—in particular, the visible HTML body the recipient will read—alongside a single legitimately signed (and ideally hidden) part, and have the entire message marked as authenticated.

CipherMail requires two conditions for the attack to apply. First, inline-PGP processing must be enabled (as noted in Section 3.2.1, it is disabled by default). Second, CipherMail’s *Add security info* feature [11] must be enabled; without it, no per-message signature indicator is delivered to the user and the situation degenerates to the Cisco and Proton Mail case. When both are enabled, CipherMail tags signed messages as `[signed]`; if mixed content is detected, it also flags them with a `[mixed content]` tag. However, the gateway is satisfied as long as the cryptographic pieces add up in aggregate, even when the message has been stitched together from several independently signed messages. On top of that, the accounting itself is loose: CipherMail merely checks that the number of valid detached signatures is at least as large as the number of unsigned MIME parts, without requiring that each unsigned part actually be covered by one of those signatures. An attacker can therefore attach multiple detached signatures over the same part, creating “signature budget” for as many additional attacker-chosen unsigned parts as desired (see Figure 7). By ensuring the parts used to create this signature budget are hidden from the user, the attacker can create an effectively arbitrary message from any sender and have it marked as properly signed; they only need access to a single signed message from the sender they wish to impersonate.

3.2.3 MIME boundary injection. Unlike MUAs, after decrypting a subset of a multipart message, a gateway must then reassemble the

parts into a MIME message before forwarding it to the recipient. As discussed in Section 2.1, for this reassembly to be safe, the gateway must ensure that the boundary strings used by the multipart MIME entities do not appear inside any of the enclosed parts. In this section, we show that most of the products we evaluate do not satisfy this requirement, and show how an attacker can exploit this to exfiltrate the plaintext of a captured ciphertext. As with the other exfiltration attacks, this requires a captured ciphertext and a victim MUA that loads remote content (CTX, MUA). To our knowledge, MIME boundary injection has not previously been described as an attack vector against email encryption.

The attack nests two multipart containers and places three encrypted parts inside the inner one (Figure 8). The first part decrypts to a closing `--sep2--` followed by a fresh `--sep1` with attacker-chosen MIME headers and the prefix of an HTML exfiltration gadget. The second part is the captured ciphertext whose plaintext the attacker wants to recover. The third part decrypts to the closing tags of the gadget. After the gateway decrypts the parts in place, the injected `--sep2--` prematurely closes the inner multipart and the injected `--sep1` opens a new outer part with the attacker-chosen `Content-Type: text/html` header. Since the inner multipart is now closed, the remaining `--sep2` become inert text inside the new HTML body, so the decrypted secret lands inside the attacker’s `src` attribute and is exfiltrated when the MUA fetches the image.

Results. The attack is not applicable to Cisco since it does not support decrypting emails with the required MIME structures due to its use of OpenSSL’s limited MIME parser [69].

SEPPmail and CipherMail decrypt the encrypted parts in place inside the original multipart container and reuse the boundary strings they find there. That container, including its boundaries, is supplied by the attacker. The attacker can, therefore, place those exact boundary strings inside the plaintexts of the encrypted parts before sending the message. By default, CipherMail only treats a part as encrypted when its filename matches a format-specific pattern (e.g. a `.pgp` suffix for inline-PGP, or `.p7m` for S/MIME), so the attacker has to label the encrypted parts accordingly.

Proton Mail is affected through a predictable-boundary variant. The Proton Mail Bridge derives the multipart boundaries by iterated SHA-256 hashing of the server-assigned message identifier. A malicious server can, therefore, compute the boundary strings and place them inside the plaintexts of the encrypted parts before sending the message.

3.3 Exploiting Bounces

Table 4: Results for bounce attacks. All exfiltrate plaintext.

	Full Message	Header Dec.	Padding Oracle
SEPPmail	✓	✓	✓
CipherMail	✓+	✓	✓
Cisco	✓+	✓	✓
Proton Mail	~	~	~

+ Gateway generates full message bounces itself.

A defining feature of SEPPmail, CipherMail and Cisco ESA is that they relocate decryption from the recipient’s MUA to the gateway,

leaving the resulting plaintext to traverse the rest of the organisation’s mail infrastructure—internal MTAs, mailbox stores, content scanners, MUAs—exactly as if it had been sent in the clear. None of those downstream components are aware that the message they are processing was, moments earlier, encrypted, so any feedback they routinely return to a sender risks becoming a decryption oracle.

The most prominent such feedback channel is the *Delivery Status Notification* (DSN) as specified in RFC 3461 [44]. This includes the *bounce* message an MTA returns when it cannot deliver a message. For such reports, the specification allows either only the headers of the original message or the original message in full to be returned. In either case it is sent to the original sender, so an attacker who positioned themselves as that sender receives back whatever portion of the (now decrypted) message the DSN includes. Bounces are easy to provoke on demand: the crudest method is to fill the recipient’s mailbox until further deliveries are rejected with a “mailbox full” error; subtler approaches exploit divergences between the gateway and the downstream MTA in what counts as a well-formed message. A gateway might, for instance, happily decrypt a message that lacks a `Date` header, while a stricter downstream MTA later rejects it as malformed and returns the (decrypted) result in the bounce.

If the MTA that generates DSNs is downstream of the gateway (or within the gateway itself), the DSNs do not concern the original wrapper email, but rather its *reconstructed* version—that is, the result of the decryption process. If the email decrypts correctly, the reconstructed email contains the plaintext body and some combination of the headers of the wrapper email and the decrypted headers, depending on gateway implementation and encryption standard. If the decryption fails, the reconstructed version is largely the same as the wrapper, but some headers may still change (e.g. to add information about the decryption error).

We identify three ways to turn DSN exposure into a decryption oracle, summarised in Table 4. All three require a captured ciphertext and the ability to read the resulting bounce messages (CTX, DSN). The first leaks the entire message in a single query assuming the bouncing MTA attaches the reconstructed message in full. The other two are malleability attacks that recover successive plaintext fragments and need only the headers of the bounced message, so they remain applicable even when the bouncing MTA omits the reconstructed body. We discuss each in turn.

Since the Proton Mail Bridge runs locally on the recipient’s machine and does not relay DSNs to external senders, none of the attacks in this section apply to it.

3.3.1 Full-message bounces. The simplest variant requires no cryptanalysis at all. If some component in the email path

- is after the decryption gateway and, therefore, receives the reconstructed message
- and is configured to or can be induced to include the original message in the DSN

then it will leak the entire reconstructed message in the DSN. This DSN is then sent to the original sender in plaintext, allowing an attacker who captured the ciphertext and can read the bounce to recover the plaintext in a single query (CTX, DSN).

Results. Cisco and CipherMail are particularly affected because their gateways, by default, include the reconstructed message in

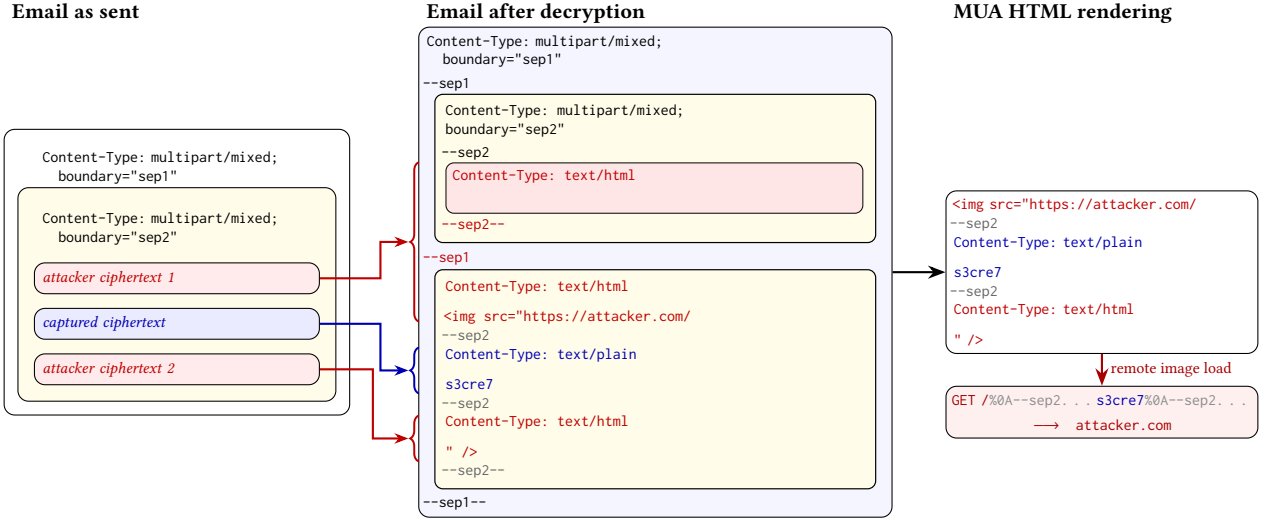


Figure 8: MIME boundary injection. After decryption, attacker-injected boundaries (red) close the inner multipart prematurely and open a new HTML part; the gateway’s original inner-multipart separators (grey) end up inside that new part as inert text, so the decrypted secret (blue) is rendered inside the attacker’s `<img>` src attribute and sent to the attacker by the MUA.

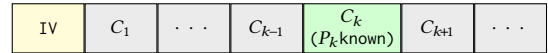
the DSNs they generate. SEPPmail only includes the headers of the reconstructed message in DSNs, but does not prevent the DSNs of downstream MTAs from leaking the entire message.

3.3.2 Header-decryption oracle. Even when a DSN omits the original body, it still includes the decrypted RFC822 header block of the reconstructed email. For unauthenticated encryption modes, an attacker can exploit the same ciphertext malleability as previously discussed in Section 3.1 to move parts of the decrypted message body into the header of the resulting decrypted email. Since this header is then included in the DSN, the attacker can still recover significant portions of the plaintext in a single query (CTX, DSN).

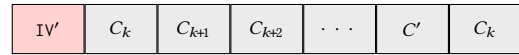
The best method to exploit this vulnerability depends on the gateway’s configuration, what parts of the message are known to the attacker, and the encryption mode used. A general approach is to use the malleability to ensure the decrypted plaintext starts with `Content-Type:` followed by the desired secret and then an empty line (see Figure 9). This ensures that the secret is interpreted as the value of the `Content-Type` header and, therefore, included in the final email’s headers. While the products differ on what other headers from the decrypted plaintext they include in the final email, they all include the `Content-Type` header as it is necessary for the recipient to interpret the message correctly.

Despite relying on ciphertext malleability, this attack also applies to emails encrypted under authenticated encryption. AES-GCM is the only authenticated encryption mode whose support is required by S/MIME [58]; an attacker cannot tamper with a GCM ciphertext in place without invalidating its integrity tag. However, they can instead induce the gateway to decrypt the same plaintext under a weaker mode. OpenSSL’s S/MIME stack, which underlies two of the gateways we evaluate, accepts AES in the (malleable) CFB mode, enabling a *backwards-compatibility* attack [36]. CFB decryption differs from GCM’s underlying CTR decryption only in what it feeds into the block cipher: the previous ciphertext block C_{i-1} in

Captured ciphertext blocks (CBC mode):



Forged ciphertext blocks (new IV, truncated content):



$$IV' = C_{k-1} \oplus P_k \oplus \text{Content-Type:}, \quad C' = C_{k-1} \oplus P_k \oplus \text{empty line}$$

Decrypted RFC822 message:

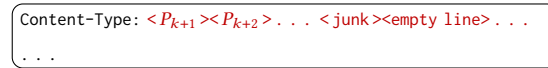


Figure 9: Header-decryption oracle.

CFB ($P_i = C_i \oplus E_k(C_{i-1})$), and the counter value CTR_i in CTR ($P_i = C_i \oplus E_k(CTR_i)$). The attacker can use this to rewrap the captured AES-GCM message as a CFB ciphertext that interleaves each GCM counter value with the corresponding ciphertext block (see Figure 10). Under CFB decryption, this rewrapped ciphertext reproduces the original plaintext blocks interleaved with garbled blocks. With careful placement and a modest amount of luck ($p \sim 1 - 2^{-4}$, depending on the MIME parser), the garbled blocks do not produce characters such as newlines, which would break MIME parsing, and the attack succeeds at recovering plaintext.

Results. Cisco ESA, SEPPmail, and CipherMail all accept the forged ciphertext and expose the chosen header to the attacker through the DSN for the reconstructed message. SEPPmail is particularly affected: rather than only including the handful of headers (most prominently `Content-Type`) strictly required for downstream delivery, it copies nearly arbitrary headers from the decrypted plaintext into the outgoing message. This gives the attacker more flexibility in crafting the attack. In the construction of Figure 9, each byte

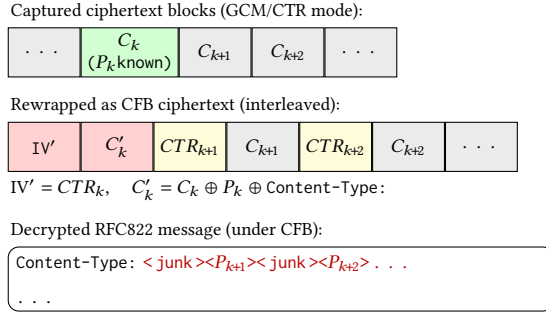


Figure 10: GCM-to-CFB downgrade for header-decryption.

of the forced prefix must be cancelled against a byte of known plaintext at the matching offset within P_k , so forcing the 14-byte Content-Type: prefix requires a correspondingly long contiguous window of known plaintext at a known offset. SEPPmail’s permissiveness instead lets the attacker substitute a shorter attacker-chosen header name such as X: that needs as few as three known bytes, extending the oracle to captured messages whose known-plaintext window is too small to cover the Content-Type prefix. In fact, it makes the required known-plaintext small enough to be feasibly brute-forceable.

Real-world Efficiency. To measure the robustness of the GCM-to-CFB variant against various MIME parsers, we implement the attack using one known plaintext block to recover one 16-byte target plaintext block with each forged email. The attack induces two junk blocks in the decryption. We empirically measured how likely these junk blocks are to prevent the attack by interfering with the MIME parsing of the SEPPmail and Cisco ESA gateways: the failure rate for a single attack attempt (decrypting a single target block) is 11% for SEPPmail and 0.07% for Cisco ESA. A failure caused by the junk block is retrievable: the attacker can choose a different valid forged-header block, changing the CFB input that produces the junk and thereby resampling it. Under a standard pseudorandomness assumption, a single-attempt failure rate of q results in a residual failure probability of q^n after n attempts. CipherMail is not affected by this downgrade because it does not support CFB mode. The variant is not applicable to the Proton Mail Bridge because it is PGP-only and does not relay external DSNs.

3.3.3 Padding oracle. Even when a DSN quotes only headers, many of the gateways we study still leak a one-bit decryption-success signal through a subject-line tag that they append to each successfully decrypted message—for example, CipherMail’s [decrypted] tag or SEPPmail’s [secure] tag. Since the tag is added to the subject, which is an email header, an attacker can obtain it from the DSN for the reconstructed message (Figure 11). Because the gateway adds this tag only after a successful decryption, it can be used to cleanly distinguish decryption successes from PKCS#7 padding failures. For AES-CBC ciphertexts, this is exactly the distinguisher required to mount a CBC padding oracle attack [73], again needing only a captured ciphertext and access to the bounce messages (CTX, DSN).

This distinguisher allows the attacker to learn one plaintext byte at the cost of (on average) 2^7 queries. While decrypting a larger

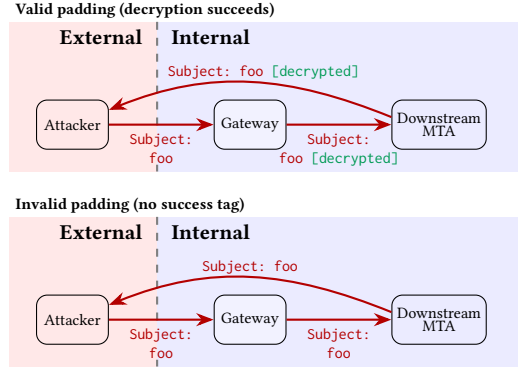


Figure 11: DSN [decrypted] tag padding oracle.

ciphertext requires many thousands or even millions of queries, using DSNs as a decryption oracle allows an attacker to perform these queries in a completely automated fashion. This significantly increases the feasibility of performing such an attack compared to equivalent attacks on MUAs that require user interaction.

As with the header-decryption oracle, a *backwards-compatibility* attack [36] can be used to extend this oracle to captured AES-GCM ciphertexts. To this end, the attacker exploits the CTR-mode (and thus GCM-mode) identity $E_k(CTR_i) = C_i \oplus P_i$. They submit a two-block CBC ciphertext ($CTR_i \oplus 0 \times 10 \dots 10$, $C_i \oplus P_{i,\text{guess}}$), the last block of which decrypts to $0 \times 10 \dots 10$ —a valid full-block padding—exactly when $P_{i,\text{guess}} = P_i$. This way, the padding oracle doubles as a plaintext-verification oracle suitable for brute-forcing low-entropy GCM ciphertext blocks. We confirmed this extension against the appliances in our lab setup.

Results. SEPPmail uses the [secure] tag to indicate successful decryption. This tag is on by default, so the oracle is available out of the box. CipherMail’s [decrypted] serves the same purpose but must be manually enabled. However, CipherMail adds custom cryptographic-status headers by default, which we use as an oracle. Cisco’s ESA does not report the cryptographic status of incoming messages by default, leaving it up to administrators to configure the gateway to handle emails differently depending on whether S/MIME processing succeeds. While this means there is no default subject-line tag that an attacker can use to detect decryption success, successful decryption still updates the Content-Type header, which can be used as an oracle. In practice, gateways or upstream MTAs may impose rate limits, greylisting, or anti-abuse mechanisms that increase the time required for a complete attack. In Appendix A.2, we quantify how such rate limits affect the cost of these attacks.

3.4 User Signalling

Encryption gateways strip the cryptographic layer before delivery. This means that the recipient’s mail client has no native signal that a message was encrypted or signed. A simple alternative employed by some gateways we study is to prepend or append a short tag string to the Subject header. This allows an attentive user to observe the cryptographic status in emails’ subject lines. For instance, we previously saw that CipherMail tags messages bearing a valid sender signature as [signed]. Since the subject line is otherwise

Table 5: User signalling results. Unicode (tag) spoofing and Bounded sani(tisation bypass) attacks set arbitrary security tags, while Tag hiding simply removes tags from the UI.

	Unicode spoof.	Bounded sani.	Tag hiding
SEPPmail	✓	✓	✓
CipherMail	✓	✓	✓
Cisco	~	~	~
Proton Mail	~	~	~

attacker-controllable, the gateway must also sanitise incoming subjects before appending its own tag: specifically, any pre-existing occurrences of the tags must be stripped. If sanitisation is incomplete, or if the attacker can conceal the gateway’s genuine tag, the recipient may be shown a false cryptographic-status signal. To the end user, this can, for example, have the same effect as forging an email’s cryptographic signature.

We consider an attacker who sends a fresh email to a recipient behind the gateway and controls only its Subject header; here we assume that DKIM is bypassed or not enforced. All three attacks in this subsection therefore require only the ability to bypass DKIM/SPF sender-authentication checks (SPF). We identify three ways in which such an attacker can defeat the sanitisation, summarised in Table 5. The first exploits Unicode-based visual ambiguities to smuggle a deceptive tag past the gateway’s sanitisation. The second exploits mistakes in how tag stripping is implemented. The third pads the subject with whitespace so the gateway’s own tag lands off-screen. We discuss each in turn.

Proton Mail does not implement subject-line tagging at all, so none of the attacks in this section apply to it. Cisco does not provide subject-line cryptographic-status tags like SEPPmail or CipherMail. Any equivalent hints must be assembled by the administrator from generic filtering and header-manipulation features, so there is no single vendor-endorsed tagging workflow whose sanitisation we can assess. Therefore, we likewise set Cisco aside for this subsection. Excluding these two products is not a positive result, however: neither gives its users a dedicated indicator of an email’s cryptographic status in the first place. By default, CipherMail has both subject-line tagging (*Add security info* [11]) and subject-line sanitisation (*Enable Subject Filter* [13]) disabled. In order to evaluate CipherMail in this section, we assume both are enabled.

3.4.1 Unicode-based tag spoofing. An attacker can craft subject lines that exploit Unicode confusables [71] and control characters to create tags that appear legitimate to users but evade the gateway’s ASCII-oriented sanitisation logic. For example, the Cyrillic small letter *le* (U+0435) is visually nearly identical to the Latin *e* (U+0065). An attacker who replaces the Latin *e* in a tag with its Cyrillic homoglyph produces a subject line that is visually indistinguishable from a genuine tag. A sanitiser which checks for the exact ASCII string would not recognise the lookalike as a tag and would pass it through unmodified, so the unencrypted message arrives with what looks like a legitimate security indicator.

Right-to-left override characters (U+202E/U+202C, as used in so-called Trojan Source attacks [6]) and zero-width-space characters bypass the sanitiser in similar ways. The homoglyph variant

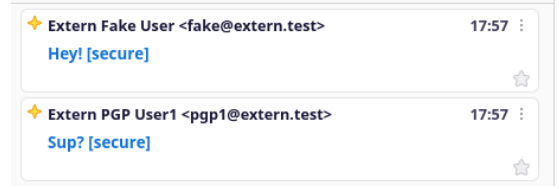


Figure 12: Forged (above) and legitimate (below) [secure] tags in Thunderbird.

remains the preferred vector, however, because some MUAs (notably Thunderbird) visibly neutralise directional and zero-width characters in the subject line by rendering them as spaces, whereas they render homoglyphs verbatim. See Appendix B for an overview of how various MUAs handle different attack variants.

3.4.2 Bounded sanitisation bypass. Gateways strip any existing occurrence of their tag patterns from the subject before adding real tags based on the cryptographic status. If this stripping is run for only a bounded number of passes (rather than to a fixed point), an attacker can *nest* the forged tag inside a larger string that, after sanitisation, yields the tag itself. For instance, CipherMail strips tags a single time. To forge CipherMail’s [signed] tag, we can use the following construct:

Before gateway: [sig[signed]ned] Meeting notes

After gateway: [signed] Meeting notes

The sanitiser matches the inner [signed] and removes it, but the remaining characters [sig and ned] concatenate into [signed], a forged tag that was never produced by the gateway. SEPPmail runs the sanitisation multiple times but gives up if the subject still contains a tag after 52 attempts to sanitise it and uses the partially sanitised subject. Furthermore, SEPPmail does not just remove the tag but replaces it with a space character. Therefore, the attacker is limited to forging a tag which contains a space such as the [signed OK] SEPPmail uses to indicate a valid signature.

3.4.3 Tag hiding. When the gateway appends its tag at the end of the subject line, an attacker can pad the original subject with a large number of whitespace characters (spaces or tabs), pushing the legitimate tag far beyond the visible area of the recipient’s mail client. For example:

Urgent: read immediately [mixed content]
200+ invisible spaces

The user sees only Urgent: read immediately in their inbox, with, depending on the MUA, little to no indication that the gateway may have tagged the message. This technique is particularly dangerous when combined with ciphertext manipulation: if the gateway appends a warning tag such as [signed INVALID] or [mixed content] in response to tampering, the attacker can suppress that warning from view. See Appendix B for an overview of how various MUAs handle different whitespace characters.

Results. Both SEPPmail and CipherMail

- (1) forward all three Unicode variants unmodified,
- (2) run a bounded stripping loop (52 iterations for SEPPmail, 1 for CipherMail), delivering the forged tag to the user,

- (3) and preserve the attacker’s whitespace padding and append their tag at the far end of the subject.

4 Mitigations

While the architectural mismatch between gateway decryption and end-to-end security cannot be fully resolved without significant changes to the architecture, several targeted mitigations can reduce the severity of the attacks we have described. We discuss them here in the order the attacks were presented in Section 3.

Restoring the cryptographic signal. A gateway that decrypts a message should re-attach an explicit, downstream-visible marker so that an EFAIL-aware MUA can re-enable the defences it would have applied to an encrypted message, such as blocking remote content and isolating decrypted parts from neighbouring HTML. However, this requires MUAs that are aware of such a marker, which requires coordination between gateway vendors and MUA vendors. Other MTAs that, for example, scan the decrypted emails for viruses would also have to honour this marker, which would then also make it a potential avenue for malicious emails to bypass such scans. To protect against direct exfiltration, vendors could re-encode emails into a safe structure, detecting EFAIL-like mixed parts and dropping them or setting them as attachments (i.e. Content-Disposition: attachment rather than inline). Protecting against gadget variants is harder: CipherMail’s EFAIL countermeasure of filtering 7-bit plaintext has a high likelihood of catching modified ciphertexts but exposes a new and more efficient side channel through bounce oracles.

In response to our disclosure, CipherMail version 6.3.3 [12] implements several heuristics to detect potential EFAIL attack emails. If any of these flag the email, the reconstructed email is stripped of references to external resources, thereby blocking the EFAIL exfiltration channel [7].

Handling mixed content. The three mixed-content attacks of Section 3.2 require separate fixes. For the inline-PGP ciphertext splicing attack, the gateway can strip any prefix and suffix around the decrypted fragment so that the resulting plaintext cannot complete attacker-chosen markup. This is the fix SEPPmail deployed in version 15.0.1 [63] in response to our disclosure. The gateway can also prevent an attacker from using HTML to automatically exfiltrate the plaintext by refusing to embed a decrypted inline-PGP fragment in such active entities. For the partial-signature attack, a message should be marked as signed only when the entire message is covered by a single verified signature. For the MIME boundary-injection attack, the standard MIME discipline dictates the fix: whenever the body of a multipart message is altered—whether by decryption, re-serialisation, or fresh assembly—the gateway must ensure that the content does not contain the boundary using any of the methods discussed in Section 2.1. To that end, both SEPPmail and CipherMail now base64-encode decrypted content [60] while Proton Mail Bridge now generates the boundary using a cryptographic PRNG [52].

Closing bounce-based oracles. The most immediate mitigation concerns full-message bounces: DSNs must not carry the decrypted body back to the external sender. Gateways should be configured to include only the delivery-status headers in the DSN. They should

also intercept and sanitise outbound DSNs before they leave the trusted perimeter. Even a header-only DSN provides an oracle through the headers it includes. Gateways should forward only a fixed allow-list of headers from the decrypted plaintext—Content-Type, MIME-Version, and all other headers needed for downstream delivery. Additionally, they should validate that each forwarded value is well-formed (for example, Content-Type should parse as a recognised MIME type). This does not eliminate the oracle, since the attacker can, for instance, still probe whether a chosen-prefix decryption parses as a legitimate header value, but it reduces the per-query bandwidth and removes the short attacker-chosen header names (e.g., X:) that make the oracle feasible against small known-plaintext windows.

Ideally the gateway would reject unauthenticated ciphertext altogether. However, RFC 8551 currently makes AES-CBC MUST-implement for S/MIME, so a gateway cannot refuse CBC without violating the standard. Where a gateway cannot reject malleable ciphertexts, it can at least refuse to distinguish padding failures from successful decryption. Some MUAs already ignore PKCS#7 padding errors on the client side.

Given how the reconstructed email may be handled by various MTAs that can behave in unknown ways, the safest approach is to ensure that no DSN leaves the organisation’s trusted email ecosystem. In order to do so, CipherMail can now rewrite the email such that any DSN is sent to a configurable email address instead of the sender of the original email [7].

Hardening subject-line tags. The signalling attacks of Section 3.4 each admit a targeted fix. The Unicode-lookalike attack can be addressed by matching tags against the Unicode Consortium’s confusables table [71] to detect homoglyphs and block that variant of the attack. This mitigation was adopted by SEPPmail in version 15.0.3 [64] and CipherMail in version 6.3.3 [12] as a result of our disclosure. Confusables alone, however, do not catch right-to-left-override or zero-width characters; those should be stripped or normalised. These variants both remained unmitigated in SEPPmail as of July 2026.

The bounded-sanitisation bypass can be prevented by running the tag stripping to a fixpoint. As a result of our disclosure, SEPPmail was updated in version 15.0.3 [64] to error out when the sanitisation loop reaches its iteration cap with a tag pattern still present, rather than silently failing and forwarding a forged tag. CipherMail version 6.3.3 [12] instead re-applies the subject filter repeatedly without any upper bound on the number of iterations.

Tag hiding can be defeated either by collapsing runs of whitespace in the subject before appending the gateway’s tag or by prepending the tag rather than appending it, so that any attacker-controlled trailing padding cannot push it past the visible window. As a result of our disclosure, SEPPmail was updated in version 15.0.3 [64] to collapse long runs of whitespace.

5 Discussion

In this paper, we shed light on an under-studied segment of cryptography “in the wild”: email security gateways.

We only explored a fraction of the market, but our findings show that concentrating all cryptographic operations in a gateway is highly problematic. The gateway becomes a single point of failure

whose compromise would have devastating consequences. From this perspective, it is notable that, in the course of our research, we even discovered a remote code execution (RCE) vulnerability in SEPPmail (CVE-2026-27441) as well as a path-traversal file-access issue (CVE-2026-27442, CWE-22), now disclosed and fixed [63], and Cisco ESA received a patch in early 2026 for an RCE vulnerability that had been exploited in the wild for over a month [17, 20]. So the risk of centralisation is not merely theoretical.

Even in the absence of software exploits, we have shown that the security promises of all of the products we analysed can be violated. This also applies to products such as the Proton Mail Bridge, where centralisation is much less of a concern, but where a malicious server model is appropriate given the E2EE security claims of the vendor. We argue here that these failings arise at least in part from the security gateway architecture itself, rather than being mere software bugs. This is underlined by the fact that the only vulnerability discussed here which we did not successfully exploit on a gateway that implemented all relevant features is the PGP gadget EFMAIL variant—the one vulnerability that we only included to complete the EFMAIL analysis rather than because we consider it an architectural issue. There are three causes for these architectural flaws.

1. Loss of Cryptographic Signal. Cryptographic security cannot be achieved in isolation. Recent attacks show that even when a system’s cryptographic core is well designed, serious vulnerabilities can emerge at the level of the overall product [2, 24, 57].

Email clients, as the final endpoint where a message is displayed, cannot be completely divorced from the underlying encryption. The EFMAIL saga itself is a prime example of this: fundamentally the attack was possible because rendering of the HTML content in emails was executed separately from their decryption; email gateways once again impose this separation, enabling EFMAIL reanimation.

The loss of cryptographic signal not only revives EFMAIL, but also introduces a non-trivial user interface problem. The email client is no longer responsible for decrypting email or verifying signatures. So how can a gateway signal to the end user that an email was encrypted, or authenticated? The products we analysed provide two possible answers. One is to simply provide no (user-visible) signal to the email client. This approach negates much of the utility of authenticating emails in the first place. The alternative is to add ‘tags’ to the subject line of the email after decryption or signature verification. We show that all the products using tags as indicators fail to protect them from adversarial influence, making tags at best an unreliable signal, and at worst a misleading one. A more robust alternative is to condition delivery on the email passing appropriate cryptographic verification. While this approach is used in some closed systems such as KIM [32], the prevalence of self-signed certificates for both PGP and S/MIME poses a major difficulty for adopting this method when interoperability is required.

2. Reconstruction Hazards. Some of the most insidious attacks we discover originate from the fact that gateways have to reconstruct emails after decryption and signature verification. This exposes the gateways to a plethora of security-critical decisions: Should inline-PGP be processed? Which new MIME boundaries should be picked? What counts as a signed message?

MUAs have faced substantial security scrutiny, revealing vulnerabilities of similar severity to the ones we discuss here [35, 45], but gateways had so far escaped academic attention. It is perhaps not surprising that we discover multiple attacks relating to mixed content, where many ambiguous edge-cases can emerge. These attacks range from EFMAIL-style direct exfiltration of inline-PGP, to signature bypasses. These are easy to mitigate, but their prevalence (affecting all the applicable products) makes them concerning.

The novel boundary injection attacks are unique to the gateway setting, since they arise when MIME separators remain attacker-known across a decrypt-and-reserialise step.

3. A Different Threat Model. Existing email encryption standards, such as S/MIME and PGP, were not designed with automated decryption in mind. Modern designs assume strong attackers with access to decryption oracles and require CCA-secure encryption, cf. the PGP cryptographic refresh [77]. By contrast, S/MIME is stuck in the past: it not only supports, but mandates, the implementation of non-AEAD cipher modes, such as plain AES-CBC, and non-CCA-secure public key encryption such as RSA-PKCS#1 v1.5. Cryptography textbooks caution against both of these schemes because of issues like padding oracle attacks [4, 73]. We have shown that S/MIME is, indeed, vulnerable to such attacks.

One reason why S/MIME may so far have escaped a cryptographic refresh is that, in the traditional end-to-end setting, padding oracle attacks are not practical: they require thousands to millions of queries, each query potentially delivering an email to a user. The attacks additionally rely on the user opening the email, and either the user or their MUA acting on the content. This is not the case for email gateways: they will automatically decrypt large numbers of emails and are part of a larger infrastructure of MTAs, forwarding emails to other servers after decryption. In that setting, each probe can yield feedback from the mail pipeline—such as delivery status notifications from a downstream MTA—without the need for any human involvement. Appendix A.2 reports per-byte wall-clock and email-volume costs for automated bounce-driven padding-oracle decryption.

These serious vulnerabilities are eclipsed in some products by an even bigger security failure: errors resulting from the MTA forwarding process can include a full decryption of the target email, which delivers plaintext straight into the hands of the adversary. This highlights, once again, the importance of a holistic security analysis of cryptographic systems.

6 Conclusion

Gateway encryption is hazardous, and the security of gateways fails in highly nontrivial ways. Bounces must be generated; subject lines must be tagged (or not); ciphertext must be decrypted; MIME boundaries must be chosen. Each of these mundane operations can become an attack vector. We proposed mitigations, but many of these are just stopgaps making exploitation harder. Looking forward, the S/MIME standard needs updating and backwards compatibility in PGP must be discarded if meaningful cryptographic security is to be achieved. EFMAIL and user signalling issues can be resolved, but it is beyond the scope of this paper to design the right protocols for gateway-to-MUA communication, and to conduct user studies on the effectiveness of different signalling mechanisms.

The 29 attacks we present here are far from being exhaustive. For example, follow-up research [75] prompted by our disclosure already reports additional findings on SEPPmail, the product that we analysed in most detail. Gateways expose a large attack surface, demonstrating that a holistic security analysis of cryptographic protocols must consider the environment in which they are deployed.

References

- [1] Martin R. Albrecht and Kenneth G. Paterson. 2024. Analysing Cryptography in the Wild - A Retrospective. *Cryptology ePrint Archive, Report 2024/532*. <https://eprint.iacr.org/2024/532>
- [2] Fabian Bäumer, Marcus Brinkmann, and Jörg Schwenk. 2024. Terrapin Attack: Breaking SSH Channel Integrity By Sequence Number Manipulation. In *USENIX Security 2024: 33rd USENIX Security Symposium*, Davide Balzarotti and Wenyan Xu (Eds.). USENIX Association, Philadelphia, PA, USA. <https://www.usenix.org/conference/usenixsecurity24/presentation/baumer>
- [3] Birk Blechschmidt and Ben Stock. 2023. Extended Hell(o): A Comprehensive Large-Scale Study on Email Confidentiality and Integrity Mechanisms in the Wild. In *USENIX Security 2023: 32nd USENIX Security Symposium*, Joseph A. Calandrino and Carmela Troncoso (Eds.). USENIX Association, Anaheim, CA, USA, 4895–4912. <https://www.usenix.org/conference/usenixsecurity23/presentation/blechschmidt>
- [4] Daniel Bleichenbacher. 1998. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. In *Advances in Cryptology - CRYPTO '98 (Lecture Notes in Computer Science, Vol. 1462)*, Hugo Krawczyk (Ed.). Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA, 1–12. doi:10.1007/BFb0055716
- [5] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, and David Cooper. 2008. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280. doi:10.17487/RFC5280
- [6] Nicholas Boucher and Ross Anderson. 2023. Trojan Source: Invisible Vulnerabilities. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 6507–6524. <https://www.usenix.org/conference/usenixsecurity23/presentation/boucher>
- [7] Martijn Brinkers. 2026. Five vulnerabilities in CipherMail Gateway: EFAIL revisited, a bounce oracle, and trust-tag spoofing. <https://www.ciphermail.com/blog/ciphermail-efail-family-vulnerabilities-2026.html> Accessed: 2026-07-30.
- [8] Lara Bruseghini, Daniel Huijgens, and Kenneth G. Paterson. 2022. Victory by KO: Attacking OpenPGP Using Key Overwriting. In *ACM CCS 2022: 29th Conference on Computer and Communications Security*, Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi (Eds.). ACM Press, Los Angeles, CA, USA, 411–423. doi:10.1145/3548606.3559363
- [9] CipherMail B.V. 2018. EFAIL: detection and prevention. <https://www.ciphermail.com/blog/efail-detection-and-prevention.html> Accessed: 2026-04-17.
- [10] CipherMail B.V. 2026. About Us. <https://www.ciphermail.com/about.html> Accessed: 2026-04-15.
- [11] CipherMail B.V. 2026. Add security info. <https://www.ciphermail.com/documentation/gateway-administration-guide/settings.html#add-security-info-recipient-p> Accessed: 2026-04-19.
- [12] CipherMail B.V. 2026. CHANGELOG.md · main · CipherMail B.V. / ciphermail-community-gateway · GitLab. https://gitlab.com/ciphermail/ciphermail-community-gateway/-/blob/main/CHANGELOG.md?ref_type=heads#633-2026-06-09 Accessed: 2026-07-30.
- [13] CipherMail B.V. 2026. Enable Subject Filter. <https://www.ciphermail.com/documentation/gateway-administration-guide/settings.html#enable-subject-filter-sender-p-recipient-p> Accessed: 2026-04-21.
- [14] CipherMail B.V. 2026. On-premises email encryption gateway. <https://www.ciphermail.com/gateway.html> Accessed: 2026-04-15.
- [15] CipherMail B.V. 2026. PGP/Inline For Incoming Email Enabled. <https://www.ciphermail.com/documentation/gateway-administration-guide/settings.html#pgp-inline-for-incoming-email-enabled-sender-p> Accessed: 2026-04-17.
- [16] CipherMail B.V. 2026. Scan HTML. <https://www.ciphermail.com/documentation/gateway-administration-guide/settings.html#scan-html-sender-p> Accessed: 2026-04-17.
- [17] Cisco Systems, Inc. 2025. UAT-9686 actively targets Cisco Secure Email Gateway and Secure Email and Web Manager. <https://blog.talosintelligence.com/uat-9686/> Accessed: 2026-04-22.
- [18] Cisco Systems, Inc. 2026. Cisco Secure Email Gateway - Cisco. <https://www.cisco.com/c/en/us/support/security/email-security-appliance/series.html> Accessed: 2026-04-15.
- [19] Cisco Systems, Inc. 2026. Cisco Secure Email Secure/Multipurpose Internet Mail Extensions Ciphertext Decryption Vulnerabilities. <https://sec.cloudapps.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-esa-smime-disc-dzw4rEdY> Accessed: 2026-09-03.
- [20] Cisco Systems, Inc. 2026. Reports About Cyberattacks Against Cisco Secure Email Gateway And Cisco Secure Email and Web Manager. <https://sec.cloudapps.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-sma-attack-N9bf4> Accessed: 2026-04-22.
- [21] Jeremy Clark, Paul C. van Oorschot, Scott Ruoti, Kent E. Seamons, and Daniel Zappala. 2021. SoK: Securing Email - A Stakeholder-Based Analysis. In *FC 2021: 25th International Conference on Financial Cryptography and Data Security, Part I (Lecture Notes in Computer Science, Vol. 12674)*, Nikita Borisov and Claudia Diaz (Eds.). Springer Berlin Heidelberg, Germany, Virtual Event, 360–390. doi:10.1007/978-3-662-64322-8_18
- [22] D. Crockier, T. Hansen, and M. Kucherawy. 2011. DomainKeys Identified Mail (DKIM) Signatures. RFC 6376. doi:10.17487/RFC6376
- [23] Luca De Feo, Bertram Poettering, and Alessandro Sorniotti. 2021. On the (In)Security of ElGamal in OpenPGP. In *ACM CCS 2021: 28th Conference on Computer and Communications Security*, Giovanni Vigna and Elaine Shi (Eds.). ACM Press, Virtual Event, Republic of Korea, 2066–2080. doi:10.1145/3460120.3485257
- [24] Nir Drucker and Shay Gueron. 2021. Selfie: reflections on TLS 1.3 with PSK. *Journal of Cryptology* 34, 3 (July 2021), 27. doi:10.1007/s00145-021-09387-y
- [25] Enlyft. 2026. Companies using Cisco Email Security and its marketshare. <https://enlyft.com/tech/products/cisco-email-security> Accessed: 2026-04-27.
- [26] European Union. 2016. Regulation (EU) 2016/679 (GDPR), Article 32 (Security of processing). <https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng> Art. 32(1)(a): “the pseudonymisation and encryption of personal data”; Art. 32(2): relevant risks include unauthorised disclosure of, or access to, personal data transmitted, stored or otherwise processed..
- [27] European Union. 2016. Regulation (EU) 2016/679 (GDPR), Recital 83. <https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng> “the controller or processor should evaluate the risks inherent in the processing and implement measures to mitigate those risks, such as encryption”; Recital 83 also refers to risks to personal data transmitted, stored or otherwise processed..
- [28] European Union. 2022. Directive (EU) 2022/2555 (NIS2), Article 21(2)(h). <https://eur-lex.europa.eu/eli/dir/2022/2555/oj/eng> Requires “policies and procedures regarding the use of cryptography and, where appropriate, encryption”..
- [29] European Union. 2024. Regulation (EU) 2024/2847 (Cyber Resilience Act), Annex I, Part I, point 1(c). <https://eur-lex.europa.eu/eli/reg/2024/2847/oj/eng> Products with digital elements shall “protect the confidentiality of stored, transmitted or otherwise processed data, personal or other, such as by encrypting relevant data at rest or in transit by state of the art mechanisms”..
- [30] Federal Republic of Germany. 2017. Federal Data Protection Act (BDSG), Section 64. https://www.gesetze-im-internet.de/englisch_bdsge/englisch_bdsge.html#p0509 Section 64(2): measures may include “pseudonymization and encryption of personal data”; Section 64(3)(8): confidentiality and integrity must be protected during transfers of personal data or transport of data media..
- [31] Ned Freed and Dr. Nathaniel S. Borenstein. 1996. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045. doi:10.17487/RFC2045
- [32] gematik GmbH. 2026. KIM: Sichere Kommunikation im Medizinwesen & Datenaustausch | gematik. <https://www.gematik.de/anwendungen/kim> Accessed: 2026-04-29.
- [33] Lexi Groves and Liam Wachter. 2025. gpg.fail: OpenPGP implementation vulnerabilities. Website; 39C3 talk “To sign or not to sign: Practical vulnerabilities in GPG & friends”. <https://gpg.fail/>
- [34] Russ Housley. 2009. Cryptographic Message Syntax (CMS). RFC 5652. doi:10.17487/RFC5652
- [35] Fabian Ising, Damian Poddebniak, Tobias Kappert, Christoph Saatjohann, and Sebastian Schinzel. 2023. Content-Type: multipart/oracle - Tapping into Format Oracles in Email End-to-End Encryption. In *USENIX Security 2023: 32nd USENIX Security Symposium*, Joseph A. Calandrino and Carmela Troncoso (Eds.). USENIX Association, Anaheim, CA, USA, 4175–4192. <https://www.usenix.org/conference/usenixsecurity23/presentation/ising>
- [36] Tibor Jager, Kenneth G. Paterson, and Juraj Somorovsky. 2013. One Bad Apple: Backwards Compatibility Attacks on State-of-the-Art Cryptography. In *ISOC Network and Distributed System Security Symposium - NDSS 2013*. The Internet Society, San Diego, CA, USA.
- [37] Kahlil Jallad, Jonathan Katz, and Bruce Schneier. 2002. Implementation of Chosen-Ciphertext Attacks against PGP and GnuPG. In *ISC 2002: 5th International Conference on Information Security (Lecture Notes in Computer Science, Vol. 2433)*, Agnes Hui Chan and Virgil D. Gligor (Eds.). Springer Berlin Heidelberg, Germany, Sao Paulo, Brazil, 90–101. doi:10.1007/3-540-45811-5_7
- [38] Jonathan Katz and Bruce Schneier. 2000. A Chosen Ciphertext Attack Against Several E-Mail Encryption Protocols. In *USENIX Security 2000: 9th USENIX Security Symposium*, Steven M. Bellovin and Greg Rose (Eds.). USENIX Association, Denver, Colorado, USA. <https://www.usenix.org/conference/9th-usenix-security-symposium/chosen-ciphertext-attack-against-several-e-mail-encryption>
- [39] Vlastimil Klima and Tomas Rosa. 2002. Attack on Private Signature Keys of the OpenPGP Format, PGP(TM) Programs and Other Applications Compatible with OpenPGP. *Cryptology ePrint Archive, Report 2002/076*. <https://eprint.iacr.org/2002/076>

- [40] Tadayoshi Kohno, Yasemin Acar, and Wulf Loh. 2023. Ethical Frameworks and Computer Security Trolley Problems: Foundations for Conversations. In *USENIX Security 2023: 32nd USENIX Security Symposium*, Joseph A. Calandrino and Carmela Troncoso (Eds.). USENIX Association, Anaheim, CA, USA, 5145–5162. <https://www.usenix.org/conference/usenixsecurity23/presentation/kohno>
- [41] Litmus Software, Inc. 2026. Email Client Market Share: Trends & Statistics 2026. <https://www.litmus.com/email-client-market-share/> Accessed: 2026-06-25.
- [42] Florian Maury, Jean-René Reinhard, Olivier Levillain, and Henri Gilbert. 2015. Format Oracles on OpenPGP. In *Topics in Cryptology – CT-RSA 2015 (Lecture Notes in Computer Science, Vol. 9048)*, Kaisa Nyberg (Ed.). Springer, Cham, Switzerland, San Francisco, CA, USA, 220–236. doi:10.1007/978-3-319-16715-2_12
- [43] Serge Mister and Robert J. Zuccherato. 2006. An Attack on CFB Mode Encryption as Used by OpenPGP. In *SAC 2005: 12th Annual International Workshop on Selected Areas in Cryptography (Lecture Notes in Computer Science, Vol. 3897)*, Bart Preneel and Stafford Tavares (Eds.). Springer Berlin Heidelberg, Germany, Kingston, Ontario, Canada, 82–94. doi:10.1007/11693383_6
- [44] Keith Moore. 2003. Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs). RFC 3461. doi:10.17487/RFC3461
- [45] 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 *USENIX Security 2019: 28th USENIX Security Symposium*, Nadia Heninger and Patrick Traynor (Eds.). USENIX Association, Santa Clara, CA, USA, 1011–1028. <https://www.usenix.org/conference/usenixsecurity19/presentation/muller>
- [46] 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. In *ACNS 2019: 17th International Conference on Applied Cryptography and Network Security (Lecture Notes in Computer Science, Vol. 11464)*, Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung (Eds.). Springer, Cham, Switzerland, Bogota, Colombia, 24–42. doi:10.1007/978-3-030-21568-2_2
- [47] Phong Q. Nguyen. 2004. Can We Trust Cryptographic Software? Cryptographic Flaws in GNU Privacy Guard v1.2.3. In *Advances in Cryptology – EUROCRYPT 2004 (Lecture Notes in Computer Science, Vol. 3027)*, Christian Cachin and Jan Camenisch (Eds.). Springer Berlin Heidelberg, Germany, Interlaken, Switzerland, 555–570. doi:10.1007/978-3-540-24676-3_33
- [48] Gurur Öndarö, Jonas Kaspereit, Samson Umezulike, Christoph Saatjohann, Fabian Ising, and Sebastian Schinzel. 2025. S/MIME: Collecting and Analyzing S/MIME Certificates at Scale. In *USENIX Security 2025: 34th USENIX Security Symposium*, Lujo Bauer and Giancarlo Pellegrino (Eds.). USENIX Association, Seattle, WA, USA, 6737–6756. <https://www.usenix.org/conference/usenixsecurity25/presentation/oendaro>
- [49] 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 *USENIX Security 2018: 27th USENIX Security Symposium*, William Enck and Adrienne Porter Felt (Eds.). USENIX Association, Baltimore, MD, USA, 549–566. <https://www.usenix.org/conference/usenixsecurity18/presentation/poddebniak>
- [50] Proton AG. 2023. There are now over 100 million Proton Accounts | Proton. <https://proton.me/blog/proton-100-million-accounts> Accessed: 2026-04-27.
- [51] Proton AG. 2026. Encrypt your desktop email app with Proton Mail Bridge | Proton. <https://proton.me/mail/bridge> Accessed: 2026-04-15.
- [52] Proton AG. 2026. Modify message building MIME boundary generation logic (stochastic). <https://github.com/ProtonMail/proton-bridge/commit/7413410d65a7c7ef685809edc258d3b53ddff27> Accessed: 2026-07-30.
- [53] Proton AG. 2026. Proton Mail: Get a free email account with privacy and encryption | Proton. <https://proton.me/mail> Accessed: 2026-04-15.
- [54] Thomas Roessler, Michael Elkins, Raph Levien, and Dave Del Torto. 2001. MIME Security with OpenPGP. RFC 3156. doi:10.17487/RFC3156
- [55] Christoph Saatjohann. 2025. KIM 1.5: Noch mehr Chaos In der Medizinischen Telematikinfrastruktur (TI). Talk at the 39th Chaos Communication Congress (39C3). <https://media.ccc.de/v/39c3-kim-1-5-noch-mehr-chaos-in-der-medizinischen-telematikinfrastruktur-ti>
- [56] Christoph Saatjohann, Fabian Ising, and Sebastian Schinzel. 2024. KIM: Chaos In der Medizin. In *Sicherheit, Schutz und Zuverlässigkeit: Konferenzband der 13. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI), Sicherheit 2024*. Gesellschaft für Informatik e.V., Worms, Germany, 93–104. doi:10.18420/sicherheit2024_006
- [57] Matteo Scarlata, Giovanni Torrisi, Matilda Backendal, and Kenneth G. Paterson. 2026. Zero Knowledge (About) Encryption: A Comparative Security Analysis of Three Cloud-based Password Managers. *IACR Cryptol. ePrint Arch.* 2026 (2026), 58. <https://eprint.iacr.org/2026/058>
- [58] Jim Schaad, Blake C. Ramsdell, and Sean Turner. 2019. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification. RFC 8551. doi:10.17487/RFC8551
- [59] Jörg Schwenk, Marcus Brinkmann, Damian Poddebniak, Jens Müller, Juraj Somorovsky, and Sebastian Schinzel. 2020. Mitigation of Attacks on Email End-to-End Encryption. In *ACM CCS 2020: 27th Conference on Computer and Communications Security*, Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna (Eds.). ACM Press, Virtual Event, USA, 1647–1664. doi:10.1145/3372297.3417878
- [60] SEPPmail AG. 2026. Prevent unsafe MIME boundary injection. <https://downloads.seppmail.com/extrelnotes/150/ERN15.0.html#prevent-unsafe-mime-boundary-injection> Accessed: 2026-07-30.
- [61] SEPPmail AG. 2026. SEPPmail - Secure e-mail communication made easy. <https://seppmail.com/> Accessed: 2026-04-15.
- [62] SEPPmail AG. 2026. SEPPmail clients. <https://www.seppmail.com/clients/> Accessed: 2026-04-27.
- [63] SEPPmail AG. 2026. SEPPmail Vulnerability Disclosure. <https://downloads.seppmail.com/extrelnotes/150/ERN15.0.html#seppmail-vulnerability-disclosure> Accessed: 2026-04-28.
- [64] SEPPmail AG. 2026. SEPPmail Vulnerability Disclosure 15.0.3. <https://downloads.seppmail.com/extrelnotes/150/ERN15.0.html#seppmail-vulnerability-disclosure-1503> Accessed: 2026-04-28.
- [65] Falko Strenzke. 2023. ForgedAttributes: An Existential Forgery Vulnerability of CMS and PKCS#7 Signatures. *Cryptology ePrint Archive*, Report 2023/1801. <https://eprint.iacr.org/2023/1801>
- [66] Andris Suter-Dörig, Matteo Scarlata, and Kenneth G. Paterson. 2026. Proof-of-Concept Implementations for Email Encryption Gateway Attacks. <https://github.com/AppliedCrypto-ETHZ/email-encryption-gateways-pocs>. Accessed: 2026-09-03.
- [67] Andris Suter-Dörig. 2026. *Breaking SEPPmail: A Case Study of Email Encryption in the Wild*. Master’s thesis. ETH Zurich. <https://ethz.ch/content/dam/ethz/special-interest/infk/inst-infsec/appliedcrypto/education/theses/msc-thesis-andris-seppmail.pdf>
- [68] Swiss Confederation. 2022. Ordinance of 31 August 2022 on Data Protection (DPO), Articles 2–3. https://www.fedlex.admin.ch/filestore/fedlex.data.admin.ch/eli/cc/2022/568/20250401/en/html/fedlex-data-admin-ch-eli-cc-2022-568-20250401-en.html#art_2 Art. 2 requires confidentiality and integrity; Art. 3 requires controls for data transmission devices and protects personal data in the event of disclosure or transport.
- [69] The OpenSSL Project Authors. 2021. SMIME_read_CMS. https://docs.openssl.org/3.6/man3/SMIME_read_CMS/ OpenSSL documentation, BUGS section: “The MIME parser used by SMIME_read_CMS() is somewhat primitive. While it will handle most S/MIME messages more complex compound formats may not work.” Accessed: 2026-04-16.
- [70] Alexander Ulrich, Ralph Holz, Peter Hauck, and Georg Carle. 2011. Investigating the OpenPGP Web of Trust. In *ESORICS 2011: 16th European Symposium on Research in Computer Security (Lecture Notes in Computer Science, Vol. 6879)*, Vijay Atluri and Claudia Diaz (Eds.). Springer Berlin Heidelberg, Germany, Leuven, Belgium, 489–507. doi:10.1007/978-3-642-23822-2_27
- [71] Unicode Consortium. 2025. Unicode Technical Standard #39: Unicode Security Mechanisms. <https://www.unicode.org/reports/tr39/#confusables> Confusables section. Accessed: 2026-04-29.
- [72] United States Department of Health and Human Services. 2003. HIPAA Security Rule, 45 C.F.R. § 164.312(e) (Transmission security). <https://www.ecfr.gov/current/title-45/subtitle-A/subchapter-C/part-164/subpart-C/section-164.312> Requires technical security measures to guard against unauthorised access to electronic protected health information transmitted over an electronic communications network; § 164.312(e)(2)(ii) provides for encryption when appropriate.
- [73] Serge Vaudenay. 2002. Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS.... In *Advances in Cryptology – EUROCRYPT 2002 (Lecture Notes in Computer Science, Vol. 2332)*, Lars R. Knudsen (Ed.). Springer Berlin Heidelberg, Germany, Amsterdam, The Netherlands, 534–546. doi:10.1007/3-540-46035-7_35
- [74] Chuhan Wang, Kaiwen Shen, Minglei Guo, Yuxuan Zhao, Mingming Zhang, Jianjun Chen, Baojun Liu, Xiaofeng Zheng, Haixin Duan, Yanzhong Lin, and Qingfeng Pan. 2022. A Large-scale and Longitudinal Measurement Study of DKIM Deployment. In *USENIX Security 2022: 31st USENIX Security Symposium*, Kevin R. B. Butler and Kurt Thomas (Eds.). USENIX Association, Boston, MA, USA, 1185–1201. <https://www.usenix.org/conference/usenixsecurity22/presentation/wang-chuhan>
- [75] Dario Weiss, Manuel Feifel, and Olivier Becker. 2026. SEPPmail Secure E-Mail Gateway: Critical RCE and LFI Vulnerabilities (CVE-2026-2743, CVE-2026-7864, CVE-2026-44127, CVE-2026-44128). InfoGuard Labs blog. https://labs.infoguard.ch/posts/seppmail_secure_email_gateway_rce_vulnerabilities_cve-2026-2743_cve-2026-7864_cve-2026-44127_cve-2026-44128/
- [76] Alma Whitten and J. Doug Tygar. 1999. Why Johnny Can’t Encrypt: A Usability Evaluation of PGP 5.0. In *USENIX Security 99: 8th USENIX Security Symposium*, G. Winfield Treese (Ed.). USENIX Association, Washington, DC, USA. <https://www.usenix.org/conference/8th-usenix-security-symposium/why-johnny-cant-encrypt-usability-evaluation-pgp-50>

[77] Paul Wouters, Daniel Huigens, Justus Winter, and Niibe Yutaka. 2024. OpenPGP. RFC 9580. doi:10.17487/RFC9580

A Evaluation

This appendix gives the experimental detail behind the padding-oracle attacks, whose practicality depends on repeated gateway interaction. It quantifies the per-byte cost of automated padding-oracle decryption.

A.1 Testing Environment

We ran the measurements in an isolated two-host lab. Both hosts ran Debian GNU/Linux 13. The attacker host generated the probes on an Intel Core i7-6700 system with 4 cores, 8 threads, and 62 GiB of RAM. The appliance host used an Intel Core i7-4790K system with 4 cores, 8 threads, and 31 GiB of RAM, and ran each gateway as a QEMU 10.0.8/KVM virtual machine with 4 vCPUs and 4096 MB of RAM. It also provided the SMTP and DNS services needed for delivery and bounces. Absolute timings should therefore be read as lab measurements rather than production throughput claims.

Attack-side tooling used OpenSSL 3.5.6 and Python 3.13.5. The lab SMTP and DNS services used Python 3.13.5 with `aiosmtpd` 1.4.6 and `dnslib` 0.9.26. The two hosts communicated over a routed link with no packet loss in short probes and an average RTT of about 11.5 ms.

A.2 Padding Oracle

This subsection quantifies the feasibility argument in Section 3.3.3: bounce-mediated padding oracles let an attacker who can capture a ciphertext and read DSNs recover plaintext byte-by-byte without the user interaction that makes the same attack impractical against MUAs. We report two per-byte costs: wall-clock time and the average number of SMTP messages the adversary injects. CipherMail and Cisco ESA match Table 9. For SEPPmail we use release **15.0.4.2**; the vulnerability analysis elsewhere used 14.0.3, on which we already demonstrated the generic oracle (Section 3.3.3).

Each trial encrypts a fresh AES-256-CBC S/MIME envelope and attempts to decrypt the full content-encryption payload. The oracle answer is learned from DSNs that quote forwarded headers: SEPPmail’s [secure] tag, CipherMail’s [decrypted] tag (with subject-line tagging enabled as in Section 3.3.3), and, on Cisco, a content-filter rule that prepends [verified] after successful S/MIME processing. All product configurations follow Section 3.3.3.

A.2.1 Generic attack. Our baseline is a product-agnostic implementation of the standard byte-at-a-time CBC padding oracle [73]. For cross-product comparability it is deliberately a *textbook* attack: it uses no assumption about character set or message structure. Such an attack needs slightly less than 129 oracle queries per plaintext byte on average. Our wall-clock measurements, by contrast, drove probes as fast as each gateway could accept them using up to five parallel SMTP connections. Because of this, the runs that produced the timing column often sent more messages per recovered byte than the theoretical average listed in the table.

Table 6: Generic padding oracle: cost per recovered plaintext byte.

Product	Time	Emails
SEPPmail	160 s	128.6
CipherMail	20 s	128.6
Cisco	3.7 s	128.6

The timing differences mostly reflect message-processing throughput in our lab setup. SEPPmail is the slowest under this attack; CipherMail and Cisco process probes substantially faster, with Cisco the fastest of the three. However, all three of them could easily recover a 16-character code within an hour.

A.2.2 Specialized variants. The generic attack in Table 6 is intentionally product-neutral. The variants below exploit specific gateway behaviours to work better in rate-limited environments.

SEPPmail (carrier batching). The specialized path multiplexes many PKCS#7 probes as `message/rfc822` parts inside a single multipart/mixed carrier email. SEPPmail forwards each of these attached S/MIME encrypted emails as a separate email. Therefore, a single carrier email yields multiple oracle probes. This reduces attacker-submitted traffic to one carrier per recovered byte, plus one additional carrier per block. In our lab this path is slower than the generic attack, and it requires SEPPmail 15.0.3 or higher: earlier releases forwarded nested S/MIME attachments with a null envelope sender (`<>`), suppressing the DSNs needed by the attacker. The measurements shown in Table 7 were run on SEPPmail 15.0.4.2.

CipherMail (7-bit bounce oracle). CipherMail’s optional post-decryption 7-bit check [9] exposes a different signal, namely the `X-Ciphermail-Info-SMIME-Illegal-Chars-Found` header. It is added whenever the decrypted plaintext contains a byte b not in the set $\{0x09, 0x0a, 0x0d\} \cup [0x20, 0x80)$. An attacker can use this more balanced signal to recover a byte in fewer than nine oracle queries on average. However, this requires using the knowledge that the initial plaintext is in this allowed range. Using this knowledge in the generic PKCS#7 padding-oracle attack also reduces the average number of emails per recovered byte to ~ 50 . The lower email count does not translate into a proportional speedup: each query depends on the previous result, so the attack cannot be pipelined like the generic padding oracle.

Cisco (connection rotation). By default, Cisco ESA allows a single SMTP connection to send at most 10 messages. The generic timings were measured with this limit disabled; the specialized timings in Table 7 instead rotate SMTP connections every 10 messages. This stays within the default connection policy and changes the measured per-byte cost only marginally.

Table 7: Specialized bounce-oracle variants: cost per recovered plaintext byte.

Product	Time	Emails
SEPPmail	250 s	1.06
CipherMail	7.1 s	8.96
Cisco	3.7 s	128.6

Limitations. The harness runs in a controlled lab, not on production paths with greylisting or vendor anti-abuse throttling. Oracle availability still depends on gateway configuration, as discussed in Section 3.3.3. The SEPPmail carrier variant is inapplicable before version 15.0.3.

B Mail client behaviour

A subset of the attacks in Section 3 rely on the recipient’s MUA rendering the final message in a particular way: the EFAIL direct-exfiltration variant (Section 3.1) requires the client to splice sibling MIME parts into a single HTML rendering context, while most of the user-signalling attacks (Section 3.4) depend on how the client displays specially crafted subjects. We therefore tested these client-dependent attacks against the three most widely used desktop mail clients [41]: Apple Mail 16.0, Microsoft Outlook (classic) 2605, and Mozilla Thunderbird 140.12, each in its default configuration. Table 8 summarises the results.

Table 8: Client-dependent attacks in the three most popular desktop mail clients.

Attack	Apple Mail	Outlook	Thunderbird
EFAIL direct exfiltration	✓	✗	✗
Homoglyph spoof	✓	✓	✓
RTL-override spoof	✓	✓	✗
Zero-width spoof	✓	✓	✗
Tag hiding (spaces)	✓	✓	✗
Tag hiding (tabs)	✗	✓*	✗

* When using tabs to hide the tag, Outlook includes a visible ellipsis to indicate the truncation.

The homoglyph spoof is the only attack that works against all three clients, making it effective independent of the MUA. While Thunderbird’s aggressive subject-normalisation prevents most of the other attacks, these results show that only addressing the homoglyphs is insufficient to prevent lookalike tags.

C Product versions

Table 9 lists the product releases we used for the analysis in this paper.

Table 9: Analysed product versions.

Product	Version	Notes
SEPPmail	15.0.4.2	Full-stack DSN tests
SEPPmail	14.0.3	All other tests
CipherMail	6.2.5	
Cisco ESA	16.0.1	
Proton Mail Bridge	3.24.2	

Table 10 lists the patches that vendors have released in response to our disclosure as of September 2026.

D Open Science

We have developed a proof-of-concept implementation for every attack reported in this paper, corresponding to each ✓ in the per-section summary tables and every attack in Table 1.

Because coordinated disclosure with the affected vendors was ongoing at the time of submission to ACM CCS 2026, we chose not to release the full proof-of-concept code at that time. As of 3 September 2026, all of our 90-day coordinated disclosure windows have elapsed, and we published the PoCs to a public repository [66].

E Ethical Considerations

Our work analyses the security of four email-encryption gateway products that process confidential and security-sensitive communications. This creates a clear need to consider possible harms arising from the research, the steps we took to mitigate them, and the benefits of publication. We formulate this as a stakeholder-based analysis and take a consequentialist viewpoint [40], focusing on the likely harms and benefits of publication for the affected parties.

The main stakeholders in our work are: the email gateway companies whose products we study; the users and organisations that rely on these products to protect email; and us, the researchers.

Concerning us as researchers, publication creates both risks and benefits. On the risk side, disclosing weaknesses in security products can create professional friction with vendors and may expose us to reputational, legal, and time costs associated with coordinated disclosure. On the benefit side, publishing the work contributes to the scientific record, helps validate the time spent on the research, and allows us to communicate the dangers of gateway-based email security to the wider community. We sought to manage these risks responsibly by verifying our findings with proof-of-concept implementations, performing experiments only in controlled settings, engaging vendors privately before wider release, and coordinating with relevant government cybersecurity centres where needed.

Concerning the email gateway companies, publication may create reputational and commercial harm, and may impose engineering costs by forcing vendors to investigate and patch the reported issues. We mitigate these harms through coordinated disclosure.

We disclosed our findings to SEPPmail on 31 October 2025. Our disclosure comprised 56 vulnerabilities identified over roughly six months of work. Only a subset is discussed here, selected for its relevance to the paper’s cryptographic focus. We proposed a standard 90-day embargo period, answered follow-up questions, and reviewed the partial fixes released in versions 15.0.1 through 15.0.3. By the end of the 90 days on 29 January 2026, many issues remained unpatched. We reported one additional finding to SEPPmail on 5 May 2026, which was patched in version 15.0.5, released 30 May 2026. On 31 July 2026, we offered to extend the embargo period and informed them that we intended to publish our work after 1 September 2026.

For Proton Mail, we disclosed on 21 April 2026, also proposing a 90-day embargo. The MIME boundary injection issue was fixed in Bridge version 3.25.0, released 1 June 2026. We informed them on 29 July 2026 that we would publish our work after 1 September 2026. As of 3 September 2026, as far as we can tell from the public source and release notes, the PGP direct exfiltration attack remains unfixed in the latest Proton Mail Bridge release (3.26.0). Proton awarded a bounty for our findings.

For CipherMail, we disclosed 12 findings. Version 6.3.3, released 9 June 2026, addressed all of them except long-subject untagging

Table 10: Vendor patches released in response to our disclosure.

Product	Version	Date	CVE	Fixed issues
SEPPmail	15.0.1	6 Jan 2026	CVE-2026-2747	Mixed ciphertext (Section 3.2.1) [63]
SEPPmail	15.0.3	3 Mar 2026	CVE-2026-29144, CVE-2026-29141, CVE-2026-29137	All user signalling vulnerabilities (Section 3.4) [64]
SEPPmail	15.0.5	30 May 2026	Not assigned	MIME boundary (Section 3.2.3) [60]
Proton Mail Bridge	3.25.0	1 Jun 2026	Not assigned	MIME boundary (Section 3.2.3) [52]
CipherMail	6.3.3	9 Jun 2026	Not assigned	All issues except long subject untagging (Section 3.4.3) [12]
Cisco ESA	—	2 Sep 2026	CVE-2026-20354, CVE-2026-20355	Published advisory [19] which we believe refers to our findings.

(Section 3.4.3) [12]; CipherMail also published a public account of the findings and their mitigations [7].

We disclosed the four Cisco ESA issues to Cisco PSIRT. Cisco published an advisory on 2 September 2026, assigning CVE-2026-20354 and CVE-2026-20355 to S/MIME handling vulnerabilities which we believe to be our findings [19].

Stepping back from the discussion of individual disclosures, our aim was not to embarrass vendors, but to give them actionable information and time to mitigate the issues while still preserving the public-interest value of eventual publication.

Concerning users, the ethical balance is the sharpest. On the one hand, publication of attack techniques can lower the barrier for misuse, and the affected systems are deployed precisely to protect sensitive communications. On the other hand, users also bear the main risk if weaknesses in these products remain undisclosed: they may place trust in cryptographic protections that, in the gateway setting, can often be bypassed or undermined. We therefore view disclosure and publication as serving users’ interests, provided that they are handled carefully. All vulnerabilities described in this paper were discovered through controlled testing on self-hosted or trial instances of the affected products; no production systems or third-party user data were involved. The test environments were isolated lab setups with self-signed certificates, dedicated MTAs, and attacker-controlled listener services on a private network segment. No messages were sent to or from real users, and no real encryption keys were compromised. The one departure from a fully isolated setup was for Proton Mail, where we used dedicated research accounts to verify that the attack behaved as expected in a realistic deployment. We also limit the immediate operational value of the paper to attackers: while the attacks are described at a level sufficient for scientific scrutiny and reproducibility, we withheld some low-level exploitation details and working exploit scripts until vendors have had a fair opportunity to respond.

On balance, we believe that publication is justified. The likely long-term benefit—improving the security of email gateway products and clarifying their actual security properties for users and operators—outweighs the short-term risks, especially given the care we took in experimentation, disclosure, and release timing.

F Generative AI Usage

Generative AI tools (large language models) were used in two capacities during this work. First, an LLM (Anthropic Opus 4.6) was used to generate initial scaffolding code for many of the proof-of-concept implementations. All generated code was subsequently reviewed,

completed, extended into a full PoC, and verified by the authors through execution against the target systems in a test environment.

Second, an LLM (Anthropic Opus 4.6) was used to expand the authors’ structured notes and bullet-point outlines into initial prose for some parts of this paper. However, several sections were written entirely by the authors without LLM assistance (Sections 5 and 6). Where LLM-generated text was used as a starting point, it was critically reviewed and often heavily revised (Sections 1.1, 2); in cases where the output was not of sufficient quality, it was discarded and the relevant section was rewritten from scratch by the authors (Sections 1, 2.3, 3, 4). OpenAI GPT 5.4, 5.5 & 5.6 and Anthropic Opus 4.7, 4.8, & 5 were used to edit for grammar, and to catch typos and inconsistencies. The research design, threat modelling, experimental methodology, and all technical findings are entirely the authors’ own work.

We took care that all interactions with LLMs used subscriptions with privacy controls, ensuring that training on user inputs was disallowed.