

Limited Additional Mechanisms for PKIX and SMIME
Internet-Draft
Intended status: Best Current Practice
Expires: 7 August 2026

D. Van Geest
CryptoNext Security
F. Strenzke
MTG AG
3 February 2026

Best Practices for Signed Attributes in CMS SignedData
draft-ietf-lamps-cms-euf-cma-signeddata-01

Abstract

The Cryptographic Message Syntax (CMS) has different signature verification behaviour based on whether signed attributes are present or not. This results in a potential existential forgery vulnerability in CMS and protocols which use CMS. This document describes the vulnerability and lists mitigations and best practices to avoid it.

About This Document

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

The latest revision of this draft can be found at <https://lamps-wg.github.io/cms-euf-cma-signeddata/draft-ietf-lamps-cms-euf-cma-signeddata.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-ietf-lamps-cms-euf-cma-signeddata/>.

Discussion of this document takes place on the Limited Additional Mechanisms for PKIX and SMIME Working Group mailing list (<mailto:spasm@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/spasm/>. Subscribe at <https://www.ietf.org/mailman/listinfo/spasm/>.

Source for this draft and an issue tracker can be found at <https://github.com/lamps-wg/cms-euf-cma-signeddata>.

Status of This Memo

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

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

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

This Internet-Draft will expire on 7 August 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
2. Conventions and Definitions	4
3. mimeType Content Type	4
4. Best Practices	5
4.1. Existing Uses of id-data	5
4.2. Recipient Verification	5
5. Mitigations	6
5.1. Recipient Detection	6
5.2. Sender Detection	6
6. Security Considerations	6
7. ASN.1 Module	8
8. IANA Considerations	9
9. References	9
9.1. Normative References	9
9.2. Informative References	10
Appendix A. RFCs Using the id-data EncapsulatedContentInfo Content Type	11
A.1. RFC 8894 Simple Certificate Enrolment Protocol	12
A.2. RFC 8572 Secure Zero Touch Provisioning (SZTP)	13
A.3. S/MIME RFCs	13
A.4. RFC 6257 Bundle Security Protocol Specification	13
A.5. RFC 5655 IP Flow Information Export (IPFIX)	13
A.6. RFC 5636 Traceable Anonymous Certificate	14
A.7. RFC 5126 CMS Advanced Electronic Signatures (CAAdES)	14
A.8. RFC 5024 ODETTE File Transfer Protocol 2	14

A.9. RFC 3126 Electronic Signature Formats for long term	
electronic signatures	14
Acknowledgments	14
Authors' Addresses	14

1. Introduction

The Cryptographic Message Syntax (CMS) [RFC5652] signed-data content type allows any number of signers in parallel to sign any type of content.

CMS gives a signer two options when generating a signature on some content:

- * Generate a signature on the whole content; or
- * Compute a hash over the content, place this hash in the message-digest attribute in the SignedAttributes type, and generate a signature on the SignedAttributes. The SignedAttributes type is placed in the signedAttrs field of the SignedData type.

The resulting signature does not commit to the presence of the SignedAttributes type, allowing an attacker to influence verification behaviour. An attacker can perform two different types of attacks:

1. Take an arbitrary CMS signed message M which was originally signed with SignedAttributes present and rearrange the structure such that the SignedAttributes field is absent and the original DER-encoded SignedAttributes appears as an encapsulated or detached content of type id-data, thereby crafting a new structure M' that was never explicitly signed by the signer. M' has the DER-encoded SignedAttributes of the original message as its content and verifies correctly against the original signature of M.
2. Let the signer sign a message of the attacker's choice without SignedAttributes. The attacker chooses this message to be a valid DER-encoding of a SignedAttributes object. The attacker can then add this encoded SignedAttributes object to the signed message and change the signed message to the one that was used to create the messageDigest attribute within the SignedAttributes. The signature created by the signer is valid for this arbitrary attacker-chosen message.

This vulnerability was presented by Falko Strenzke to the LAMPS working group at IETF 121 [LAMPS121] and is detailed in [Str23].

Section 5.3 of [RFC5652] states:

signedAttrs is a collection of attributes that are signed. The field is optional, but it MUST be present if the content type of the EncapsulatedContentInfo value being signed is not id-data.

Thus, if a verifier accepts a content type of id-data in the EncapsulatedContentInfo type when used in SignedData, then a SignerInfo within the SignedData may or may not contain a signedAttrs field and will be vulnerable to this attack. On the other hand, if the verifier doesn't accept a content type of id-data, the sender always adds the signedAttrs field, and the recipient verifies that signedAttrs is present, the attack will not succeed.

The limited flexibility of either the signed or the forged message in either attack variant may mean the attacks are only narrowly applicable. Nevertheless, due to the wide deployment of the affected protocols and the use of CMS in many proprietary systems, the attacks cannot be entirely disregarded.

As a mitigation, this document defines the new mimeType content type to be used in new uses of the CMS SignedData type when the encapsulated content is MIME encoded and thus avoid the use of the id-data content type. This document further describes best practices and mitigations that can also be applied to those protocols or systems that continue to use the content type id-data.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. mimeType Content Type

The following object identifier identifies the mimeType content type:

```
id-mime-data OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) ct(1)
  TBD2 }
```

The mimeType content type is intended as a replacement for the id-data content type in new uses of the CMS SignedData type where the content is MIME encoded.

4. Best Practices

This section describes the best practices to avoid the vulnerability at the time of writing.

New uses of the CMS SignedData MUST NOT use the id-data EncapsulatedContentInfo content type. If the new content is MIME encoded, the mimeType content type SHOULD be used.

4.1. Existing Uses of id-data

When a protocol which uses the id-data EncapsulatedContentInfo content type within SignedData is updated, it SHOULD deprecate the use of id-data and use a different (new or existing) identifier. A partial list of such identifiers is found in the "CMS Inner Content Types" IANA subregistry within the "Media Type Sub-Parameter Registries". If the existing content is MIME encoded, the mimeType content type SHOULD be used. Updated protocols that do not deprecate the use of id-data SHOULD provide a rationale for not doing so. Note that accepting the content type id-data during verification is sufficient for a vulnerability to surface. Hence the measures described in Section 4.2 must be adhered to.

When a protocol uses the id-data EncapsulatedContentInfo content type within SignedData, it SHOULD specify that the signedAttrs field is either always required or always forbidden. If a protocol makes such a requirement, a recipient MUST check whether the signedAttrs field is present or absent as specified by the protocol, and fail processing if the appropriate condition is not met.

4.2. Recipient Verification

When a recipient receives a CMS SignedData, it SHOULD be checked that the EncapsulatedContentInfo content type value is the one expected by the protocol and fail processing if it is not.

As specified in Section 5.3 of [RFC5652], a SignerInfo signedAttrs field MUST be present if the content type of the EncapsulatedContentInfo value being signed is not id-data. To avoid the attack described in Section 1, when a recipient receives a CMS SignedData and the EncapsulatedContentInfo content type is not id-data, it SHOULD verify both that the expected content type was received and that each SignerInfo contains the signedAttrs field, and fail processing if either of these conditions is not met.

5. Mitigations

This section describes mitigations for cases where the best practices given above cannot be applied. When the id-data EncapsulatedContentInfo content type is used, the following mitigations MAY be applied to protect against the vulnerability described in Section 1.

5.1. Recipient Detection

This mitigation is performed by a recipient when processing SignedData.

If signedAttrs is not present, check if the encapsulated or detached content is a valid DER-encoded SignedAttributes structure and fail if it is. The mandatory contentType and messageDigest attributes, with their respective OIDs, should give a low probability of a legitimate message which happens to look like a DER-encoded SignedAttributes structure being flagged.

However, a malicious party could intentionally present messages for signing that are detected by the countermeasure and thus introduce errors into the application processing that might be hard to trace for a non-expert.

5.2. Sender Detection

This mitigation is performed by a sender who signs data received from a 3rd party (potentially an attacker).

If the sender is signing 3rd party content and will not be setting the signedAttrs field, check that the content is not a DER-encoded SignedAttributes structure, and fail if it is. Note that also in this case, a malicious party could intentionally present messages that trigger this countermeasure and thereby trigger hard-to-trace errors during the signing process.

6. Security Considerations

The vulnerability is not present in systems where the use of signedAttrs is mandatory, as long as recipients enforce the use of signedAttrs. Some examples where the use of signedAttrs is mandatory are SCEP, Certificate Transparency, RFC 4018 firmware update, German Smart Metering CMS data format. Any protocol that uses an EncapsulatedContentInfo content type other than id-data is required to use signed attributes. However, this security relies on a correct implementation of the verification routine that ensures the correct content type and presence of signedAttrs.

When the message is signed and then encrypted, though in the general case this will make it difficult for the attacker to learn the signature, the vulnerability might still be present if mitigations are not applied:

- * Signing and encryption might not be done on the same endpoints, in which case an attacker between the endpoints might be able to learn the signature for which it could remove or add the signedAttrs.
- * IND-CPA encryption does not give theoretical guarantees against an active attacker and thus does not guarantee that an attacker cannot rearrange the structure.

Conceivably vulnerable systems:

- * Unencrypted firmware update denial of service
 - Secure firmware updates often use signatures without encryption. If the forged message can bring a device, due to lack of robustness in the parser implementation, into an error state, this may lead to a denial of service vulnerability. The possibility of creating a targeted exploit can be excluded with great certainty in this case due to the lack of control the attacker has over the forged message.
- * Dense message space
 - If a protocol has a dense message space, i.e. a high probability that the forged message represents a valid command or the beginning of a valid command, then, especially if the parser is permissive with respect to trailing data, there is a risk that the message is accepted as valid. This requires a protocol where messages are signed but not encrypted.
- * Signing unstructured data
 - Protocols that sign unencrypted unstructured messages, e.g. tokens, might be affected in that the signature of one token might result in the corresponding forged message being another valid token.
- * External signatures over unstructured data

- The probably strongest affected class of systems would be one that uses external signatures, i.e. CMS signatures with absent content (that may be transmitted encrypted separately) over unstructured data, e.g. a token of variable length. In that case the attacker could create a signed data object for a known secret.

* Systems with permissive parsers

- In addition to potential issues where the protocol parser is permissive (e.g. with respect to trailing space), if the CMS parser is permissive (e.g. allows non-protocol content types, or allows missing signedAttrs with content types other than id-data) then this could result in accepting invalid messages.

Further note that it is generally not good security behaviour to sign data received from a 3rd party without first verifying that data. Section 5.2 describes just one verification step that can be performed, specific to the vulnerability described in Section 1.

7. ASN.1 Module

<CODE STARTS>

```
MimeData-2026 { iso(1) member-body(2) usa(840)
    rsadsi(113549) pkcs(1) pkcs9(9) smime(16) modules(0)
    id-mod-mime-data-2026(TBD1) }
DEFINITIONS EXPLICIT TAGS ::= BEGIN

IMPORTS
    CONTENT-TYPE
    FROM CryptographicMessageSyntax-2010
    { iso(1) member-body(2) us(840) rsadsi(113549)
      pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) }
    ;

id-mime-data OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) ct(1)
    TBD2 }

ct-MimeData CONTENT-TYPE ::= { IDENTIFIED BY id-mime-data }

END

<CODE ENDS>
```


8. IANA Considerations

In the "SMI Security for S/MIME Module Identifier" registry, create a new entry to point to this document.

Decimal	Description	Reference
TBD1	id-mod-mime-data-2026	[[This Document]]

Table 1

In the "SMI Security for S/MIME Content Types" registry, add a new entry for id-mime-data that points to this document.

Decimal	Description	Reference
TBD2	id-mime-data	[[This Document]]

Table 2

In the table "CMS Inner Content Types" add a new entry:

Name	Object Identifier	Reference
mimeData	1.2.840.113549.1.9.16.1.TBD2	[[This Document]]

Table 3

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/rfc/rfc2119>>.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<https://www.rfc-editor.org/rfc/rfc5652>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.

9.2. Informative References

- [LAMPS121] Strenzke, F., "EUF-CMA for CMS SignedData", 6 November 2024, <<https://datatracker.ietf.org/meeting/121/materials/slides-121-lamps-cms-euf-cma-00>>.
- [RFC2633] Ramsdell, B., Ed., "S/MIME Version 3 Message Specification", RFC 2633, DOI 10.17487/RFC2633, June 1999, <<https://www.rfc-editor.org/rfc/rfc2633>>.
- [RFC3126] Pinkas, D., Ross, J., and N. Pope, "Electronic Signature Formats for long term electronic signatures", RFC 3126, DOI 10.17487/RFC3126, September 2001, <<https://www.rfc-editor.org/rfc/rfc3126>>.
- [RFC3851] Ramsdell, B., Ed., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, DOI 10.17487/RFC3851, July 2004, <<https://www.rfc-editor.org/rfc/rfc3851>>.
- [RFC5024] Friend, I., "ODETTE File Transfer Protocol 2.0", RFC 5024, DOI 10.17487/RFC5024, November 2007, <<https://www.rfc-editor.org/rfc/rfc5024>>.
- [RFC5126] Pinkas, D., Pope, N., and J. Ross, "CMS Advanced Electronic Signatures (CAvES)", RFC 5126, DOI 10.17487/RFC5126, March 2008, <<https://www.rfc-editor.org/rfc/rfc5126>>.
- [RFC5636] Park, S., Park, H., Won, Y., Lee, J., and S. Kent, "Traceable Anonymous Certificate", RFC 5636, DOI 10.17487/RFC5636, August 2009, <<https://www.rfc-editor.org/rfc/rfc5636>>.
- [RFC5655] Trammell, B., Boschi, E., Mark, L., Zseby, T., and A. Wagner, "Specification of the IP Flow Information Export (IPFIX) File Format", RFC 5655, DOI 10.17487/RFC5655, October 2009, <<https://www.rfc-editor.org/rfc/rfc5655>>.
- [RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010, <<https://www.rfc-editor.org/rfc/rfc5751>>.

- [RFC6257] Symington, S., Farrell, S., Weiss, H., and P. Lovell, "Bundle Security Protocol Specification", RFC 6257, DOI 10.17487/RFC6257, May 2011, <<https://www.rfc-editor.org/rfc/rfc6257>>.
- [RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <<https://www.rfc-editor.org/rfc/rfc8551>>.
- [RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <<https://www.rfc-editor.org/rfc/rfc8572>>.
- [RFC8894] Gutmann, P., "Simple Certificate Enrolment Protocol", RFC 8894, DOI 10.17487/RFC8894, September 2020, <<https://www.rfc-editor.org/rfc/rfc8894>>.
- [Str23] Strenzke, F., "ForgedAttributes: An Existential Forgery Vulnerability of CMS Signatures", 22 November 2023, <<https://eprint.iacr.org/2023/1801>>.

Appendix A. RFCs Using the id-data EncapsulatedContentInfo Content Type

This appendix lists RFCs which use the id-data content type in EncapsulatedContentInfo. It is a best-effort list by the authors at time of authorship. The list can be used as a starting point to determine if any of BCPs in this document can be applied.

The following table summarizes the RFCs' usages of signed attributes.

RFC	Signed Attributes Usage
[RFC8894]	Requires the used of signed attributes
[RFC8572]	Says nothing about signed attributes
[RFC8551]	RECOMMENDS signed attributes
[RFC6257]	Forbids signed attributes
[RFC5751]	RECOMMENDS signed attributes
[RFC5655]	Says nothing about signed attributes
[RFC5636]	Forbids signed attributes
[RFC5126]	Requires signed attributes
[RFC5024]	Says nothing about signed attributes
[RFC3851]	RECOMMENDS signed attributes
[RFC3126]	Requires signed attributes
[RFC2633]	RECOMMENDS signed attributes

Table 4: RFCs using id-data

An RFC requiring or forbidding signed attributes does not necessarily mean that a recipient will enforce this requirement when verifying, their CMS implementation may simply process the message whether or not signed attributes are present. If one of the signed attributes is necessary for the recipient to successfully verify the signature or to successfully process the CMS data then the vulnerability will not apply; at least not when assuming the signer is well-behaved and always signs with signed attributes present in accordance with the applicable specification.

A.1. RFC 8894 Simple Certificate Enrolment Protocol

Figure 6 in Section 3 of [RFC8894] specifies id-data as the EncapsulatedContentInfo content type, and shows the use of signedAttrs. The document itself never refers to signed attributes, but instead to authenticated attributes and an authenticatedAttributes type. Errata ID 8247 clarifies that it should be "signed attributes" and "signedAttrs".

Since SCEP requires the use of signedAttrs with the id-data EncapsulatedContentInfo content type, and the recipient must process at least some of the signed attributes, it is not affected by the vulnerability.

A.2. RFC 8572 Secure Zero Touch Provisioning (SZTP)

Section 3.1 of [RFC8572] allows the use of the id-data content type, although it also defines more specific content types. It does not say anything about signed attributes.

A.3. S/MIME RFCs

[RFC8551], [RFC5751], [RFC3851], and [RFC2633] require the use of the id-data EncapsulatedContentInfo content type.

Section 2.5 of [RFC8551] says:

Receiving agents MUST be able to handle zero or one instance of each of the signed attributes listed here. Sending agents SHOULD generate one instance of each of the following signed attributes in each S/MIME message:

and

Sending agents SHOULD generate one instance of the signingCertificate or signingCertificateV2 signed attribute in each SignerInfo structure.

So the use of signed attributes is not an absolute requirement.

A.4. RFC 6257 Bundle Security Protocol Specification

Section 4 of [RFC6257] says:

In all cases where we use CMS, implementations SHOULD NOT include additional attributes whether signed or unsigned, authenticated or unauthenticated.

It does not specify what the behaviour should be if signed attributes are found by the receiver.

A.5. RFC 5655 IP Flow Information Export (IPFIX)

[RFC5655] is a file format that uses CMS for detached signatures. It says nothing about the use of signed attributes.

A.6. RFC 5636 Traceable Anonymous Certificate

Appendix C.1.2 of [RFC5636] says:

The signedAttr element MUST be omitted.

It does not specify what the behaviour should be if signed attributes are found by the receiver.

A.7. RFC 5126 CMS Advanced Electronic Signatures (CADES)

Section 4.3.1 of [RFC5126] specifies mandatory signed attributes.

One of the signed attributes is used to determine which certificate is used to verify the signature, so CaDES is not affected by the vulnerability.

A.8. RFC 5024 ODETTE File Transfer Protocol 2

[RFC5024] uses the id-data EncapsulatedContentInfo content type and says nothing about signed attributes.

A.9. RFC 3126 Electronic Signature Formats for long term electronic signatures

Section 6.1 of [RFC3126] requires the MessageDigest attribute, which is a signed attribute.

Acknowledgments

The authors would like to thank Russ Housley, Carl Wallace, and John Preu Mattsson for their valuable feedback on this document.

Authors' Addresses

Daniel Van Geest
CryptoNext Security
Email: daniel.vangeest@cryptonext-security.com

Falko Strenzke
MTG AG
Email: falko.strenzke@mtg.de