Technical Specification

ISO/TS 32004

First edition
2024-04

Document management — Portable Document Format — Integrity protection in encrypted documents in PDF 2.0

Gestion des documents — Format de document portable — Protection de l'intégrité dans les documents chiffrés en PDF 2.0

Reference number
ISO/TS 32004:2024(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

ISO draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents. ISO shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 171, *Document management applications*, Subcommittee SC 2, *Document file formats, EDMS systems and authenticity of information*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

When encrypting documents, it is important to not only preserve the confidentiality of the encrypted material, but also to ensure that the receiving party can verify its integrity. Encryption mechanisms defined in ISO 32000-2:2020 currently only provide confidentiality without this authentication aspect.

This document describes a mechanism to protect the integrity of an encrypted PDF document using a Message Authentication Code (MAC), with key material derived from the file encryption key. Message authentication codes are distinct from digital signatures based on public-key cryptography. Digital signatures and message authentication codes have different but complementary security properties: a valid MAC created following this document proves knowledge of the file encryption key, whereas digital signatures as defined in ISO 32000-2:2020 do not have that property.

The MAC mechanism described in this document is backwards compatible with ISO 32000-2:2020 and can also be used in PDF documents containing digital signatures.

This document follows the lexical conventions regarding the usage of bold and italics which are specified in ISO 32000-2:2020, Clause 4.

Document management — Portable Document Format — Integrity protection in encrypted documents in PDF 2.0

1 Scope

This document specifies how to extend the ISO 32000-2:2020 specification by adding extensions to the **Encrypt** dictionary and trailer dictionary to provide integrity protection to the encrypted PDF document. This document also ensures that extensions are fully backward-compatible.

These extensions are intended for developers of software that creates PDF files (PDF writers), software that reads existing PDF files and (usually) interprets their contents for display (PDF readers), software that reads and displays PDF content and interacts with the computer users to possibly modify and save the PDF file (interactive PDF processors) and PDF products that read and/or write PDF files for a variety of other purposes (PDF processors).

NOTE PDF writers and PDF readers are more specialized classifications of interactive PDF processors and both are PDF processors.

This document does not specify the following:

— specific processes for converting paper or electronic documents to the PDF file format;

— specific technical design, user interface implementation, or operational details of rendering;

— specific physical methods of storing these documents such as media and storage conditions;

— methods for validating the conformance of PDF files or PDF processors;

— required computer hardware and/or operating system.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 32000-2:2020, *Document management — Portable document format — Part 2: PDF 2.0*

IETF RFC 2104, *HMAC: Keyed-Hashing for Message Authentication. [online]. 1997.*[1)]

IETF RFC 4231, *Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512. [online]. 2005*[2)]

IETF RFC 3394, *Advanced Encryption Standard (AES) Key Wrap Algorithm. [online]*[3)]

IETF RFC 5652:2009, *Cryptographic Message Syntax (CMS). [online]. 2009*[4)]

IETF RFC 5869, *HMAC-based Extract-and-Expand Key Derivation Function (HKDF). [online]. 2010*[5)]

1) https://tools.ietf.org/html/rfc2104.html

2) https://tools.ietf.org/html/rfc4231.html

3) https://tools.ietf.org/html/rfc3394.html

4) https://tools.ietf.org/html/rfc5652.html

5) https://tools.ietf.org/html/rfc5869.html

IETF RFC 6211, *Cryptographic Message Syntax (CMS): Algorithm Identifier Protection Attribute. [online]. 2011*[6]

NIST Computer Security Objects Register (CSOR). [online]. 2009.[7]

3 Terms and Definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
PDF
Portable Document Format
file format defined by ISO 32000-2:2020

3.2
MAC
Message Authentication Code
cryptographic check sum on data that uses a symmetric key to detect both accidental and intentional modification of data

[SOURCE: ISO 16609:2022, 3.10]

3.3
file encryption key
key used for document-level encryption by a PDF security handler of version 5 or higher

Note 1 to entry: General provisions about PDF security handlers can be found in ISO 32000-2:2020, 7.6.3.

Note 2 to entry: Extensions to PDF can define security handlers other than those specified in ISO 32000-2:2020. File encryption keys defined by such security handlers are also covered by this definition.

3.4
signed revision
initial or incremental revision of a *PDF* (3.1) file that adds a digital signature or document time stamp signature to the document

3.5
unsigned revision
initial or incremental revision of a *PDF* (3.1) file that does not add a digital signature or document time stamp signature to the document

Note 1 to entry: A PDF document can contain both signed and unsigned revisions.

3.6
Abstract Syntax Notation One
ASN.1
International Standard for representing data types and structures

Note 1 to entry: The encoding rules for this abstract syntax notation are defined in ISO/IEC 8825-1.

[SOURCE: ISO 17261:2012, 3.5, modified — ISO/IEC 8825-2 has been replaced by ISO/IEC 8825-1 in the Note 1 to entry.]

6) https://tools.ietf.org/html/rfc6211.html

7) https://csrc.nist.gov/projects/computer-security-objects-register

3.7
distinguished encoding rules
DER
encoding rules that may be applied to values of types defined using the *ASN.1* (3.6) notation

[SOURCE: ISO/IEC 18014-2:2021, 3.22, modified — note 1 to entry has been removed]

3.8
hashed message authentication code
HMAC
mechanism for message authentication using a cryptographic hash function in combination with a shared secret key

Note 1 to entry: This definition has been adapted from RFC 2104.

4 Extension schema details

The developer extensions dictionary in Table 1 shall be part of the document's extensions dictionary (ISO 32000-2:2020, 7.12). It shall be included as an array entry under the **ISO_** prefix.

Encrypted PDF documents making use of the extension specified in this document shall conform to ISO 32000-2:2020.

Table 1 — Extension schema entries

Key	Type	Value
Type	name	*DeveloperExtensions*
BaseVersion	name	*2.0*
ExtensionLevel	integer	32004
ExtensionRevision	text string	:2024 NOTE The COLON (U+003A) character is part of the revision identifier.
URL	string	https://www.iso.org/standard/45877.html

NOTE Per ISO 32000-2:2020, 7.6.2, as modified by 5.2.2 in this document, the values of the **ExtensionRevision** **URL** entries are serialized in encrypted form. Table 1 lists the unencrypted values of those entries.

5 Proposed changes

5.1 Encrypt dictionary

5.1.1 Additions to ISO 32000-2:2020, 7.6.2

The content of Table 2 is appended to ISO 32000-2:2020, Table 20.

Table 2 — Additions to ISO 32000-2:2020, Table 20

Key	Type	Value
KDFSalt	byte string	*(Conditionally required; shall be a direct object)* A 32-byte salt value for use in key derivation (see 6.4). This entry is required in documents that make use of PDF MAC.

NOTE The value of the **KDFSalt** entry is intended to remain constant throughout all incremental updates of the document.

5.1.2 Additions to ISO 32000-2:2020, 7.6.4.2

The content of Table 3 is appended to ISO 32000-2:2020, Table 22.

Table 3 — Additions to ISO 32000-2:2020, Table 22

Bit position	Meaning
13	When zero, indicates that a PDF MAC token is required to be present in all revisions of the document. The location of the PDF MAC token is indicated by the **AuthCode** dictionary (see 5.2.3).

NOTE 1 This addition supersedes the provision of ISO 32000-2:2020, 7.6.4.2 requiring that PDF readers ignore all flags other than those at bit positions 3, 4, 5, 6, 9, 10, 11 and 12.

NOTE 2 The intention behind this permission bit is to signal to a PDF processor reading the document that a PDF MAC token is expected. The encrypted **Perms** entry provides a degree of tamper-resistance and helps to protect the document against attackers stripping the MAC. This protection is not without limitations: unless bit 13 is zero in *all* revisions, it can be trivially bypassed by a knowledgeable adversary.

5.1.3 Additions to ISO 32000-2:2020, 7.6.5.2

The content of Table 4 is appended to ISO 32000-2:2020, Table 24.

Table 4 — Additions to ISO 32000-2:2020, Table 24

Bit position	Meaning
13	When zero, indicates that a PDF MAC token is required to be present in all revisions of the document. The location of the PDF MAC token is indicated by the **AuthCode** dictionary (see 5.2.3).

NOTE This addition supersedes the provision of ISO 32000-2:2020, 7.6.5.2 requiring that PDF readers ignore all flags other than those at bit positions 2, 3, 4, 5, 6, 9, 10, 11 and 12.

5.2 File trailer

5.2.1 Additions to ISO 32000-2:2020, 7.5.5

The content of Table 5 is appended to ISO 32000-2:2020, Table 15.

Table 5 — Additions to ISO 32000-2:2020, Table 15

Key	Type	Value
AuthCode	dictionary	*(Required if the document is encrypted with user access permissions bit 13 zero. Shall be a direct object; PDF 2.0)* Describes a PDF MAC token to validate the integrity of an encrypted document (see 5.2.3). If present, the value of the **V** entry in the document's **Encrypt** dictionary shall be at least 5.

5.2.2 Additions to ISO 32000-2:2020, 7.6.2

The following entry is added to the bulleted list in ISO 32000-2:2020, 7.6.2.

— Any byte strings representing the value of the **MAC** key in an **AuthCode** dictionary.

NOTE This prevents strings containing PDF MAC tokens from being encrypted.

5.2.3 AuthCode dictionary

The **AuthCode** dictionary, defined in Table 6, contains a PDF MAC token or describes where to find it. All **AuthCode** dictionary entries defined below shall be direct objects, with the exception of the **SigObjRef** entry.

Table 6 — Entries in an AuthCode dictionary

Key	Type	Value
MACLocation	name	*(Required; shall not be an indirect reference)* Indicates where to find the PDF MAC token (see 6.5). The following values are defined: — *Standalone*: The DER-encoded PDF MAC token is given by the value of the **MAC** entry. — *AttachedToSig*: The PDF MAC token is an unsigned attribute on a digital signature, contained in the signature dictionary referenced by the **SigObjRef** entry. Any other values shall be second-class names (see ISO 32000-2:2020, Annex E).
ByteRange	array	*(Conditionally required; shall not be an indirect reference)* An array of four nonnegative integers describing the exact byte range over which to compute the document digest. This entry shall be present if **MACLocation** is *Standalone*. If **MACLocation** is *AttachedToSig*, this entry shall not be present. The first array element shall be zero. If the byte range is given by [0, L1, S, L2], then the document digest is computed by hashing L1 bytes starting from position 0 in the PDF file, followed by L2 bytes starting from position S in the PDF file. The uncovered region in between shall be occupied by the value of the **MAC** entry. In particular, S shall always be greater than L1. NOTE 1 This definition is (mutatis mutandis) the same as that of the **ByteRange** entry in a signature dictionary (see ISO 32000-2:2020, 12.8.1, Table 255).
MAC	byte string	*(Conditionally required; shall not be an indirect reference)* If **MACLocation** is *Standalone*, the value of this entry shall be a DER-encoded PDF MAC token, encoded as a hexadecimal byte string object. Space for the MAC string shall be allocated before it is computed. The first byte of the allocated region shall be a LESS-THAN SIGN (3Ch), the final byte shall be a GREATER-THAN SIGN (3Eh) and the sequence of bytes in between shall consist of hexadecimal digits representing the DER-encoded PDF MAC token, free of any trailing data. NOTE 2 This provision is stricter than the general syntax requirements for hexadecimal strings in ISO 32000-2:2020, 7.3.4.3, which permit white space characters and allow the final digit of the string to be dropped if it equals zero. These conventions do not apply to PDF MAC tokens. If **MACLocation** is *AttachedToSig*, this entry shall not be present.
SigObjRef	indirect reference	*(Conditionally required)* If **MACLocation** is *AttachedToSig*, this entry shall be an indirect reference to the dictionary containing the signature on which the PDF MAC token appears as an unsigned attribute. If **MACLocation** is *Standalone*, this entry shall not be present.

NOTE 3 It can be useful to contrast the lack of padding in the **MAC** entry with the situation for signature containers in ISO 32000-2:2020, 12.8.3.3.1. Since the size of a signature container is not always predictable in advance, it is common to pad the DER encoding of the signature container with zero bytes to fill the allocated region in the PDF file. On the other hand, since this document only considers MAC schemes with a fixed output size and disallows unauthenticated attributes in PDF MAC tokens (see 6.3.7), the length of the **MAC** entry can be predicted exactly.

NOTE 4 The value of the **SigObjRef** entry is identical to the value of the **V** entry in the corresponding signature form field. It exists purely to aid the validator in finding the PDF MAC token in a signed document.

6 Composing PDF MAC tokens

6.1 General

This clause defines the CMS structure of a PDF MAC token, and describes the cryptographic operations needed to produce such tokens. An ASN.1 module definition is provided in Annex A. Annex B contains information on how to validate PDF MAC tokens and Annex C lists a number of examples.

NOTE The procedure to compose and embed a PDF MAC token closely parallels the way CMS digital signatures and time stamp tokens are used in PDF (see ISO 32000-2:2020, 12.8.3.3 and 12.8.5).

This clause specifies lists of supported algorithms in various locations, with reference to the documents in which they are identified. When using these algorithms in a PDF MAC token, they shall be identified using the object identifiers in the cited references.

EXAMPLE The AES-256 key wrapping algorithm without padding would be identified by the object identifier 2.16.840.1.101.3.4.1.45.

6.2 *PdfMacIntegrityInfo* data type

The ASN.1 type *PdfMacIntegrityInfo* is defined by the following ASN.1 structure.

```
PdfMacIntegrityInfo::= SEQUENCE {
    version INTEGER,
    -- Included to facilitate future changes; shall be 0.

    dataDigest OCTET STRING,
    -- Always computed over relevant PDF ByteRange.

    signatureDigest [0] IMPLICIT OCTET STRING OPTIONAL
    -- Included only when part of a signed container.
}
```

When used as the content type of a CMS *ContentInfo* or *EncapsulatedContentInfo* structure, this data type shall be identified by the following object identifier.

```
id-ct-pdfMacIntegrityInfo OBJECT IDENTIFIER::=
    { iso(1) standard(0) iso32004(32004) pdfmac(1) 0 }
```

6.3 CMS structure of a PDF MAC token

6.3.1 General

PDF MAC tokens are CMS *ContentInfo* objects; their creation and validation shall be carried out in accordance with the provisions of RFC 5652. The content type of a PDF MAC token is *id-ct-authData*; see RFC 5652:2009, section 9.1. Additionally, the *AuthenticatedData* structure that constitutes a PDF MAC token shall satisfy the criteria enumerated in this clause.

6.3.2 Encapsulated content info of a PDF MAC token

The *contentType* of the *encapContentInfo* field shall be *id-ct-pdfMacIntegrityInfo*, and its *eContent* field shall be a DER-encoded *PdfMacIntegrityInfo* object (see 6.2).

6.3.3 Recipient info object, MAC key generation and key encryption

There shall be exactly one *recipientInfo* field, which shall have *PasswordRecipientInfo* as its type; see RFC 5652:2009, section 6.2.4. The *keyDerivationAlgorithm* of the recipient info object shall be *pdfMacWrapKdf* (see 6.4). The *keyEncryptionAlgorithm* shall be a symmetric key wrapping algorithm. The algorithm-key length combinations listed in Table 7 shall be supported in accordance with RFC 3394.

NOTE 1 A key wrapping algorithm does not only provide confidentiality, but also authenticates the encrypted key payload.

The MAC key used in the MAC algorithm shall be generated using a strong random number generator. The length of the generated MAC key shall be appropriate for the MAC algorithm being used, and should follow established guidelines for the chosen MAC scheme. The MAC key shall be encrypted and stored in the recipient info object as defined in RFC 5652:2009, section 6.2.4, using the key encryption key obtained from *pdfMacWrapKdf* (see 6.4).

Table 7 — Key wrapping algorithms

Description	KEK length (bits)	Reference
AES-256 key wrapping without padding	256	RFC 3394

NOTE 2 The KEK length column in Table 7 specifies the bit length of the key encryption key (KEK). In principle, this has no bearing on the possible lengths of the key being wrapped.

6.3.4 Digest algorithm identification

The digest algorithm identified by the *digestAlgorithm* field of the *AuthenticatedData* object shall be one of the algorithms enumerated in Table 8, in accordance with NIST CSOR. This algorithm is used to digest the **ByteRange** and (if applicable) the signature in the signed revision (see 6.6), but also to compute the *messageDigest* attribute (see 6.3.6.3).

Table 8 — Digest algorithms

Description	Reference
SHA-256	NIST CSOR
SHA-384	NIST CSOR
SHA-512	NIST CSOR
SHA3–256	NIST CSOR
SHA3–384	NIST CSOR
SHA3–512	NIST CSOR

6.3.5 MAC algorithm identification

The algorithm identifier in the *macAlgorithm* field shall identify the MAC algorithm used to compute the value of the *mac* field. The algorithm-key length combinations in Table 9 shall be supported in accordance with RFC 2104 and RFC 4231.

Table 9 — MAC algorithms

Description	Key length (bits)	Reference
HMAC with SHA-256	256	RFC 2104; RFC 4231

6.3.6 Authenticated attributes

6.3.6.1 General

The *authAttrs* field shall be present in the *AuthenticatedData* object. Some authenticated attributes are subject to additional conditions.

Implementations and future extensions may use additional authenticated attributes, provided that their usage does not conflict with the provisions of RFC 5652, RFC 6211 and this document.

6.3.6.2 Content type attribute

Exactly one attribute of type *id-contentType* shall be present, which shall have the value *id-ct-pdfMacIntegrityInfo*.

NOTE This a consequence of the fact that RFC 5652:2009, section 11.1 requires the content type of the encapsulated content (see 6.3.2) to be reflected in the authenticated attributes.

6.3.6.3 Message digest attribute

Exactly one attribute of type *id-messageDigest* shall be present, the value of which shall be an octet string obtained by hashing the *encapContentInfo eContent* octet string value using the message digest identified by the *digestAlgorithm* of the *AuthenticatedData* object.

NOTE This is a requirement from RFC 5652:2009, section 9. The tag and length octets on the *eContent* field are not part of the message digest, as specified in RFC 5652:2009, sections 9.2 and 11.2.

6.3.6.4 CMS algorithm protection attribute

Exactly one attribute of type *id-aa-CMSAlgorithmProtection* should be present, as specified in RFC 8933:2020, section 4. This attribute shall be used as specified in RFC 6211, with *digestAlgorithm* and *macAlgorithm* set to the same values as the enveloping *AuthenticatedData* structure.

6.3.7 Unauthenticated attributes

CMS unauthenticated attributes shall not be used; the *unauthAttrs* field shall be absent.

NOTE The restriction on unauthenticated attributes serves two purposes: to ensure that the size of the CMS container is always known before the MAC is computed, and to ensure that there are no unauthenticated parts of the CMS structure that can be manipulated without affecting validation after the MAC has been applied. This reduces the potential for implementation errors that lead to vulnerabilities.

6.4 Key derivation function

The key derivation function *pdfMacWrapKdf* is identified in ASN.1 by the following object identifier.

```
id-kdf-pdfMacWrapKdf OBJECT IDENTIFIER::=
    { iso(1) standard(0) iso32004(32004) pdfmac(1) 1 }
```

There are no algorithm parameters encoded in the algorithm identifier record.

The key derivation function *pdfMacWrapKdf* is an instance of the HMAC-based extract-and-expand key derivation function (HKDF) defined in RFC 5869. The parameters for HKDF, enumerated in RFC 5869, shall be taken as follows:

— the *hash algorithm* shall be SHA-256;

— the *output length* shall be the key encryption key length expected by the key wrapping algorithm (see Table 7);

— the *salt* shall be the 32-byte value stored in the **KDFSalt** field of the document's encryption dictionary;

— the *info* parameter shall be set to the 6-character string 'PDFMAC', encoded in UTF-8.

The key encryption key used to wrap the MAC key (see 6.3.3) shall be obtained by applying HKDF with these parameters. The HKDF calculation shall be carried out as specified in RFC 5869. The input key material for the HKDF evaluation shall be the file encryption key used by the document's security handler for document encryption. When using a public-key security handler (ISO 32000-2:2020, 7.6.5) with subfilter *adbe.pkcs7.s5*, the key shall be sourced from the crypt filter named **DefaultCryptFilter**.

NOTE 1 Since this document does not support security handlers of version 4 and earlier and crypt filters are mandatory in security handlers of version 5 (see ISO 32000-2:2020, 7.6.2, Table 20), this requirement covers all compatible uses of public-key security handlers enumerated in ISO 32000-2:2020.

NOTE 2 The security of a PDF MAC token hinges on the confidentiality of the input key material (i.e. the file encryption key) in this procedure. This implies that, for the purpose of MAC generation and validation, there is no meaningful distinction between owner-level access and user-level access to the PDF document (compared to ISO 32000-2:2020, 7.6.4.1). For example, adding a PDF MAC token to an encrypted PDF document with an empty user password offers no additional protection.

6.5 Location of PDF MAC tokens

6.5.1 Location of a PDF MAC token in an unsigned revision

When authenticating an unsigned revision of a PDF document, the PDF MAC token shall be written to the **MAC** entry of the **AuthCode** dictionary in the document trailer, as described in Table 6. The value of the **MACLocation** entry shall be *Standalone*.

If the most recent revision in a document with a PDF MAC token is an unsigned revision, then the **ByteRange** of the **AuthCode** dictionary shall cover the entire PDF file (except for the value of the **MAC** entry itself).

6.5.2 Location of a PDF MAC token in a signed revision

When authenticating a signed revision of a PDF document, adding the MAC to an **AuthCode** dictionary is potentially problematic, because it affects signature coverage. In such cases, the PDF MAC token shall be affixed to the digital signature or document time stamp signature as an unsigned attribute.

More precisely, the digital signature or document time stamp signature in the revision shall have an unsigned attribute with type identified by the following object identifier.

```
id-attr-pdfMacData OBJECT IDENTIFIER::=
    { iso(1) standard(0) iso32004(32004) pdfmac(1) 2 }
```

The attribute shall have exactly one value, with the following ASN.1 type:

```
PdfMacData::= ContentInfo
```

The value is a PDF MAC token subject to the provisions of 6.3.

In a signed revision with a PDF MAC token, the **AuthCode** dictionary in the document trailer shall have a **MACLocation** entry with value *AttachedToSig*, and a **SigObjRef** entry pointing to the signature dictionary containing the signature on which the PDF MAC token appears as an unsigned attribute. If the most recent revision in a document with a PDF MAC token is a signed revision, then the **ByteRange** of the relevant signature shall cover the entire PDF file (except for the value of the **Contents** entry in the signature dictionary).

6.6 Computing the digests in a PDF MAC token

6.6.1 General

As specified in 6.3, the encapsulated content (authenticated payload) of a PDF MAC token is an ASN.1 value of type *PdfMacIntegrityInfo*, defined in 6.2.

The *version* field in *PdfMacIntegrityInfo* shall be set to 0. A *PdfMacIntegrityInfo* object contains either one or two digest fields: a *dataDigest* field and (conditionally) a *signatureDigest* field. The way they are computed depends on whether the PDF MAC token appears in an unsigned or in a signed revision. The digest computations for these fields shall be carried out using the digest algorithm identified by the *digestAlgorithm* field in the PDF MAC token's *AuthenticatedData* structure.

NOTE 6.6.2 and 6.6.3 describe how the digests in a *PdfMacIntegrityInfo* object are computed. The computation of the *messageDigest* attribute is separate from that.

6.6.2 PDF MAC digests in unsigned revisions

In a PDF MAC token that appears in an unsigned revision of a PDF document, the *signatureDigest* field shall be absent. The value of the *dataDigest* field shall be obtained by digesting the bytes of the PDF document indicated by the **ByteRange** entry of the **AuthCode** dictionary in which it appears.

6.6.3 PDF MAC digests in signed revisions

In a PDF MAC token that appears in a signed revision of a PDF document, the *signatureDigest* field shall be present. Its value shall be computed by digesting the value of the *signature* field in the *SignerInfo* object on which the PDF MAC token is an unsigned attribute. The value of the *dataDigest* field shall be obtained by digesting the bytes of the PDF document indicated by the **ByteRange** entry of the signature dictionary in which it appears.

NOTE 1 The procedure for calculating the *signatureDigest* is analogous to the computation of the message imprint for a signature time stamp attribute.

NOTE 2 In a signed revision, the value of the *dataDigest* field ordinarily coincides with the value of the *messageDigest* attribute of the underlying *SignerInfo* object when the signature and MAC token use the same digest function. The *PdfMacIntegrityInfo* structure includes *dataDigest* and *signatureDigest* fields so that the PDF MAC can be validated independently of the underlying signature. This means that a PDF MAC token can protect both the integrity of the document and that of the signature, irrespective of the validation status of said signature.

Annex A
(informative)

ASN.1 module for PDF MAC

```
PDFMACSyntax { iso(1) standard(0) iso32004(32004) modules(0) pdfmac(1) }

DEFINITIONS IMPLICIT TAGS::=
BEGIN
-- EXPORTS All

IMPORTS

    ContentInfo
      FROM CryptographicMessageSyntax2004
           { iso(1) member-body(2) us(840) rsadsi(113549)
             pkcs(1) pkcs-9(9) smime(16) modules(0) cms-2004(24) };

iso32004-pdfmac OBJECT IDENTIFIER::=
    { iso(1) standard(0) iso32004(32004) pdfmac(1) }

-- PdfMacIntegrityInfo content type for PDF MAC tokens

PdfMacIntegrityInfo::= SEQUENCE {
    version INTEGER,
    dataDigest OCTET STRING,
    signatureDigest [0] IMPLICIT OCTET STRING OPTIONAL
}

id-ct-pdfMacIntegrityInfo OBJECT IDENTIFIER::= { iso32004-pdfmac 0 }

-- Identifier for KDF used to derive PDF MAC KEK

id-kdf-pdfMacWrapKdf OBJECT IDENTIFIER::= { iso32004-pdfmac 1 }

-- Unsigned attribute type for use in documents with
-- digital signatures

PdfMacData::= ContentInfo

id-attr-pdfMacData OBJECT IDENTIFIER::= { iso32004-pdfmac 2 }

END
```

Annex B
(informative)

Validation of document integrity using PDF MAC

B.1 General

This annex describes validation aspects to consider when assessing whether a PDF MAC token adequately protects a given document.

For the purposes of this annex, only the **AuthCode** dictionary in the current (most recent) revision of the document is taken into account.

The following aspects of PDF MAC validation are distinguished.

— The document binding validation step verifies the binding between the PDF MAC token and the PDF document content (possibly together with a signature). This step is described in B.2.

— The CMS validation step verifies the integrity of the PDF MAC token itself. With the exception of the way the key material is sourced, this step largely reduces to the provisions of RFC 5652:2009, 9.3. It is further described in B.3.

NOTE In particular, the CMS validation step does not distinguish between standalone PDF MAC tokens and PDF MAC tokens attached to signatures.

Both steps are crucial for establishing the validity of a PDF MAC token.

B.2 Document binding validation

B.2.1 General

Document binding validation refers to the aspects of PDF MAC validation that do not concern the internal structure of the token's CMS object, but rather the relationship between the PDF MAC token and the document in which it is embedded. This includes>

— locating a PDF MAC token in a PDF file;

— validating coverage requirements; and

— verifying whether the digest values in the token's *PdfMacIntegrityInfo* payload are correct.

The location of the PDF MAC token can be inferred from the **MACLocation** entry in the **AuthCode** dictionary in the document trailer (see 5.2.3 and 6.5).

— If the **MACLocation** is *Standalone*, the content of B.2.2 applies.

— If the **MACLocation** is *AttachedToSig*, the content of B.2.3 applies.

Absence of the **MACLocation** entry is indicative of an invalid MAC. If the **MACLocation** entry has any value other than *Standalone* or *AttachedToSig*, the result is implementation-dependent.

B.2.2 Document binding for standalone PDF MAC tokens

B.2.2.1 Locating the MAC

The validator can retrieve the PDF MAC token by parsing the **MAC** entry in the **AuthCode** dictionary.

B.2.2.2 PDF MAC coverage

The **ByteRange** entry in the **AuthCode** dictionary indicates the portion of the document protected by the PDF MAC token.

For validation purposes, the extent of the **ByteRange** entry is relevant: failure to cover the entire PDF file (excluding the value of the **MAC** entry) can be an indication that the document was tampered with. It can also indicate that the PDF MAC token has become stale, e.g. because another PDF processor failed to apply a new MAC after incrementally updating the document.

B.2.2.3 External digest

The binding between the PDF MAC token and the document can be verified by digesting the portion of the document indicated by the **ByteRange** entry (as specified in 6.6.2) and comparing the result to the *dataDigest* entry in the *PdfMacIntegrityInfo* object.

B.2.3 Document binding validation for PDF MAC tokens attached to a signature

B.2.3.1 Locating the MAC

The validator can locate the signature dictionary in which the PDF MAC token appears by following the indirect reference in the *SigObjRef* entry of the **AuthCode** dictionary. After parsing the **Contents** entry of the signature dictionary, the PDF MAC token can be retrieved from the signature's unsigned attributes (see 6.5.2).

B.2.3.2 PDF MAC coverage

The coverage of the signature's **ByteRange** also affects PDF MAC validation. This is similar to the standalone case; the only difference is that that the **ByteRange** is stored in the signature dictionary instead of the **AuthCode** dictionary.

B.2.3.3 External digests

As in the standalone case, the binding between the PDF MAC token and the document can be verified by digesting the portion of the document indicated by the signature's **ByteRange** and comparing the result to the *dataDigest* entry in the *PdfMacIntegrityInfo* object. Additionally, the binding between the PDF MAC token and the underlying digital signature can be verified by digesting the raw signature bytes and comparing the result to the *signatureDigest* entry in the *PdfMacIntegrityInfo* object. These procedures are specified in 6.6.3.

NOTE The mechanism described in this document was designed to allow PDF MAC tokens to be validated independently of the signature to which they are attached. Validation concerns for signatures are not addressed in this document. For provisions specific to PAdES signatures, see ISO 32000-2:2020, 12.8.3.4.5 and ETSI EN 319 102–1.

B.3 CMS MAC validation

The procedure for MAC validation in CMS is specified in RFC 5652:2009, 9.3 and 11.2. This document prescribes the key derivation function *pdfMacWrapKdf* (defined in 6.4) to derive the key decryption key for this procedure from the PDF file's file encryption key.