INTERNATIONAL STANDARD

ISO/IEC 19566-4

First edition
2020-03

Information technologies — JPEG systems —

Part 4: Privacy and security

Reference number
ISO/IEC 19566-4:2020(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

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 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).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see http://patents.iec.ch).

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 Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

A list of all parts in the ISO/IEC 19566 series can be found on the ISO website.

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

This document contributes to the specification of system-level functionalities. In particular, it specifies functionalities to provide a degree of trust while sharing image content and metadata, which simultaneously also allow the signalling of the associated access policies.

A huge number of images are distributed over the internet on a daily basis. For social media alone, this already accounts for over 3 billion pictures. These photos are often shared with many people without protection for personal information or access control. In addition, many portable devices have communication functionality that allows for the immediate distribution of photos after capturing them. In combination with the potential inclusion of GPS information in the file format, for example, the photo might expose private and geo-location information to the world.

In order to avoid such undesirable situations, the framework in this document provides protection mechanisms to the JPEG family of standards. For instance, encryption can be used to protect image data

The particular focus of this document is to provide codestream and file format syntax support to enable security and privacy functionality for JPEG standards, not only in support of ISO/IEC 10918-1, but also for standards such as ISO/IEC 15444 and ISO/IEC 18477.

Information technologies — JPEG systems —

Part 4:
Privacy and security

1 Scope

This document specifies privacy and security features which contribute to a system layer for JPEG standards. It defines generic structures that can be applied in all JPEG box-based file formats. In particular, this document specifies a signalling syntax supporting privacy and security features. The framework in this document is backwards-compatible with existing JPEG standards.

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.

ITU-T T.81 | ISO/IEC 10918-1, *Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines*

ISO/IEC 18477-3, *Information technology — Scalable compression and coding of continuous-tone still images box file format*

ISO/IEC 10646, *Information technology — Universal coded character set (UCS)*

ISO/IEC 18033-3, *Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers*

ISO/IEC 19566-5, *Information technology — JPEG systems — Part 5: JPEG universal metadata box format*

3 Terms and definitions

3.1 Definitions

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

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

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

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

3.1.1
box
binary structure that encapsulates an object embedded in a file

[SOURCE: ISO/IEC 19566-5:2019, 3.1.1]

3.1.2
codestream
sequence of bits representing a compressed image and associated metadata

[SOURCE: ISO/IEC 19566-5:2019, 3.1.2]

3.1.3
decrypted content
resulting data after applying the signalled decryption process on the encrypted content

3.1.4
master image
image file in which the *box* (3.1.1) is embedded

3.1.5
JPEG-1 image
image encoded in JPEG file format

Note 1 to entry: The image shall be encoded in compliance to ISO/IEC 10918-1.

3.1.6
box-based file format
file format whose composing elements are boxes containing structured data in compliance with ISO-based media file format

3.1.7
JPEG XT image
image encoded in the JPEG XT file format

Note 1 to entry: the images shall be encoded in compliance to ISO/IEC 18477.

3.1.8
Replacement Data box
sequence of *boxes* (3.1.1) of any type embedded in a JUMBF Replacement box

3.2 Abbreviated terms

JPEG	joint photographic experts group
JUMBF	JPEG universal metadata box format
P&S	privacy and security
ROI	region of interest
TOG	toggles

4 Conventions

4.1 Conformance language

In this document, the following verbal forms are used:

— “shall” indicates a requirement;

— “should” indicates a recommendation;

— “may” indicates a permission;

— “can” indicates a possibility or a capability.

Information marked as “NOTE” is intended to assist the understanding or use of the document. “Notes to entry” used in Clause 3 provide additional information that supplements the terminological data and can contain provisions relating to the use of a term.

The keyword "reserved" indicates a provision that is not specified at this time, shall not be used, and may be specified in the future. The keyword "forbidden" indicates "reserved" and in addition indicates that the provision will never be specified in the future.

4.2 Naming conventions for numerical values

Integer numbers are expressed as bit patterns, hexadecimal values, or decimal numbers. Bit patterns and hexadecimal values have both a numerical value and an associated particular length in bits.

Hexadecimal notation, indicated by prefixing the hexadecimal number by "0x", may be used instead of binary notation to denote a bit pattern having a length that is an integer multiple of 4. For example, 0x41 represents an eight-bit pattern having only its second most significant bit and its least significant bit equal to 1. Numerical values that are indicated as "binary" are bit pattern values (specified as a string of digits equal to 0, 1 or x in which the left-most bit is considered the most-significant bit and 'x' means either 0 or 1). Other numerical values not prefixed by "0x" are decimal values. When used in expressions, a hexadecimal value is interpreted as having a value equal to the value of the corresponding bit pattern evaluated as a binary representation of an unsigned integer (i.e., as the value of the number formed by prefixing the bit pattern with a sign bit equal to 0 and interpreting the result as a two's complement representation of an integer value). For example, the hexadecimal value 0xF is equivalent to the 4-bit pattern '1111' and is interpreted in expressions as being equal to the decimal number 15.

4.3 Boxes and superboxes

Annex A shall be used for the specification of boxes. The details for embedding boxes in specific file formats are defined in the particular specifications, for example, ISO/IEC 15444-1 for JPEG 2000, ISO/IEC 18477-3 for JPEG-1 and JPEG XT or the more generic ISO/IEC 14496-12 ISO base media file format (ISOBMFF).

In general, each object in the file is encapsulated within a binary structure called a box. A box that only contains other boxes is called a superbox. The binary structure is given in Figure 1.

Figure 1 — Binary structure of a box

— **LBox**: Box length. This field specifies the length of the box, stored as a 4-byte big-endian unsigned integer. This value includes all of the fields of the box, including the length and type. If the value of this field is 1, then the XLBox field shall exist and the value of that field shall be the actual length of the box. If the value of this field is 0, then the length of the box was not known when the LBox field was written. In this case, this box contains all bytes up to the end of the file. If a box of length 0 is contained within another box (its superbox), then the length of that superbox shall also be 0. This means that this box is the last box in the file. The values 2-7 are reserved for ITU-T | ISO use.

— **TBox**: Box type. This field specifies the type of information found in the Payload Data field. The value of this field is encoded as a 4-byte big-endian unsigned integer. However, boxes are generally referred to by an ISO/IEC 646 character string translation of the integer value. For all box types defined within this document, box types will be indicated as both character string and as 4-byte hexadecimal integers. Also, a space character is shown in the character string translation of the box type as "\040". All values of TBox not defined within this document are reserved for ITU-T | ISO/IEC use.

— **XLBox**: Box extended length. This field specifies the actual length of the box if the value of the LBox field is 1. This field is stored as an 8-byte big-endian unsigned integer. The value includes all of the fields of the box, including the LBox, TBox and XLBox fields.

— **Payload Data**: Box contents. This field contains the actual information contained within this box. The format of the box contents depends on the box type and will be defined individually for each type.

4.4 Graphical descriptions

Box definitions contain graphical description figures to illustrate the structure of the box. These figures should be interpreted as follows:

— The figures do not include box type and length fields.

— A sequence of rectangles is used to indicate the remaining fields of the box and their order.

— The width of the rectangle indicates the length of the field, a square rectangle indicates a 16-bit field.

— A grey background indicates a variable length field.

— Optional fields have a dashed border.

Figure 2 shows an illustrative example of a box with four fields:

— A: 8-bit required field;

— B: 16-bit required field;

— C: variable length required field;

— D: optional 32-bit field.

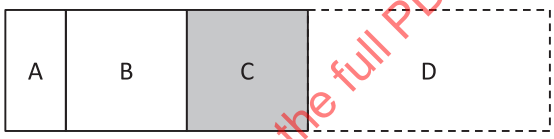


Figure 2 — Box with four fields

5 Organization of the document

This document is organized as follows:

— Annex A specifies the mechanisms for content protection and replacement to support privacy and security features.

— Annex B provides several illustrative usage examples.

Annex A
(normative)

Content protection and replacement

A.1 General

This annex specifies mechanisms for content protection and replacement. Content protection and replacement can be combined to support partial image protection. This annex builds upon the JPEG universal metadata box format (JUMBF), hence all implementations shall be compliant with ISO/IEC 19566-5.

A.2 Content protection

A.2.1 General

Content protection enables the protection of parts of an image file by encryption. When decoding an image file with protected content, the protected content shall be decrypted using the signalled encryption method, provided that the encryption method is supported by the decoder and authorization is granted. In case the protected content cannot be decrypted, the entire protected content shall be ignored. The process is illustrated in Figure A.1.

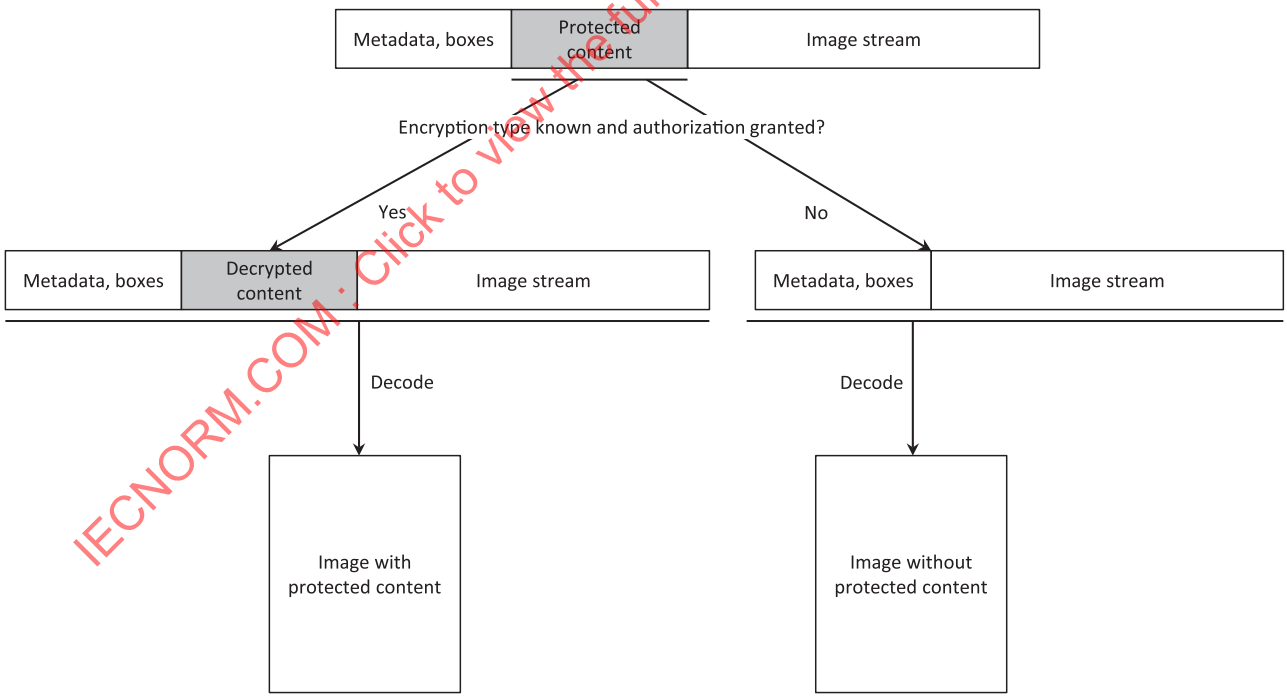


Figure A.1 — Content protection decoding process

The protected content is embedded in the master image file as a JUMBF box with Content Type "Protection", as specified in A.2.2.

A.2.2 JUMBF Protection box

The JUMBF Protection box is a JUMBF box with Content Type Protection (UUID: 74B11BBF-F21D-4EEA-98C1-0BEBF23AEFD3). The structure of the JUMBF Protection box is given in Figure A.2.

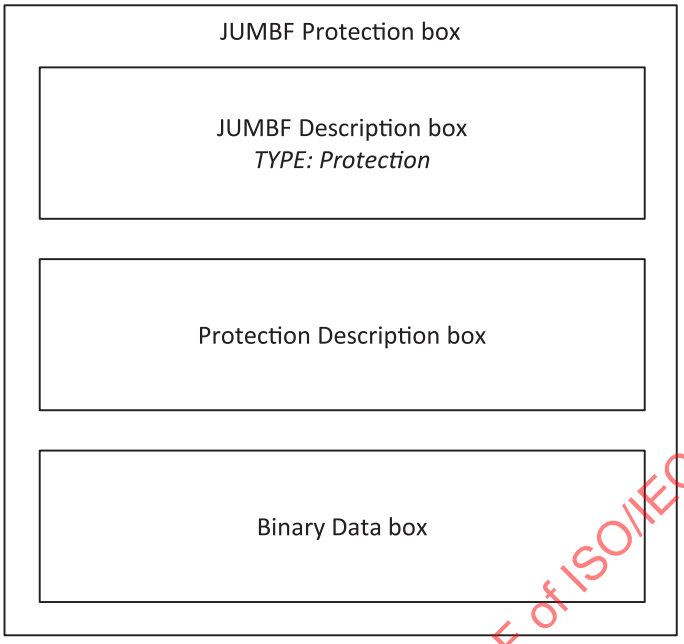


Figure A.2 — Structure of the JUMBF Protection box

In addition to the JUMBF Description box, the JUMBF Protection box contains a Protection Description box and one Binary Data box. The Binary Data box encapsulates data which is encrypted according to the scheme signalled in the Protection Description box. The resulting decrypted content shall encompass any binary data, provided that when the entire JUMBF Protection box is replaced with the decrypted content, the resulting image file shall be a valid image according to the specification of the master image. More specifically, the decrypted content can be one instance or a sequence of APP marker segments (JPEG-1 and JPEG XT only) or boxes.

A.2.3 Protection Description box

The Protection Description box signals additional information about the protected content.

The type of a Protection Description box shall be 'pspd' (0x7073 7064). The binary structure of the Protection Description box is illustrated in Figure A.3, the field values are summarized in Table A.2.

Figure A.3 — Structure of the Protection Description box

— **METHOD**: The values of the METHOD parameter and corresponding meanings are given in Table A.1.

Table A.1 — METHOD field specification

METHOD binary value	Meaning	Details
000x 0000	External	Information about the used encryption method is provided in a referenced JUMBF box. The METHOD field shall be followed by the ENC LABEL field that contains the label of the JUMBF box as null terminated ISO/IEC 10646 characters in the UTF-8 encoding.
000x 0001	AES 256 CBC	The content of the Binary Data box is encrypted using AES 256 using the CBC cipher mode (in accordance with ISO/IEC 18033-3).
000x 0010	AES 256 CBC with IV	The content of the Binary Data box is encrypted using AES 256 using the CBC cipher mode (in accordance with ISO/IEC 18033-3). The Initialisation Vector (IV) is signalled after the METHOD field and optional AR LABEL field in the 128-bit length IV field.
0001 xxxx	access rules	Access rules are provided in a referenced JUMBF box. The METHOD field, and optionally the ENC LABEL, are followed by an AR LABEL field. The AR LABEL (for referencing the access rules) shall contain the label of a JUMBF box as null terminated ISO/IEC 10646 characters in the UTF-8 encoding.
All other values are reserved for future use.		

— **ENC LABEL**: optional label referencing to a JUMBF Box that signals the used encryption method of the data in the Binary Data box. Its presence is indicated by the value of the METHOD field as specified in Table A.1.

— **AR LABEL**: optional label referencing to a JUMBF Box that signals the access policy rules of the data in the Binary Data box. Its presence is indicated by the value of the METHOD field as specified in Table A.1.

— **IV**: initialisation vector used to encrypt the data in the Binary Data box. Its presence is indicated by the value of the METHOD field as specified in Table A.1.

Table A.2 — Format of the contents of the Protection Description box

Parameter	Size (bits)	Value
METHOD	8	See Table A.1
ENC LABEL	variable	Optional null terminated UTF-8 string.
AR LABEL	variable	Optional null terminated UTF-8 string.
IV	128	Optional 128 bit initialisation vector

A.2.4 Binary Data box

The Binary Data box encapsulates binary data. The type of the binary data shall be implied by its context, otherwise the binary data shall be ignored. For example, in the context of the JUMBF Protection box the type of binary data is encrypted data as signalled in the Protection Description box. The type of a Binary Data box shall be 'bidb' (0x6269 6462). The contents of the box shall be as in Figure A.4, the fields are summarized in Table A.3.

Figure A.4 — Organization of the contents of the Binary Data box

— **DATA**: This field contains binary data.

Table A.3 — Format of the contents of the Binary Data box

Field name	Size (bits)	Value
Data	Variable	Variable length binary data

A.2.5 Decoding procedure

When a JUMBF Protection box is encountered, the encapsulated encrypted content shall be decrypted using the signalled encryption method, provided that the encryption method is supported by the decoder and authorization is granted. If the encrypted data is successfully decrypted, the resulting decrypted content replaces the entire JUMBF Protection box. In case the data cannot be decrypted, the entire JUMBF Protection box is ignored. Thereafter, in both cases decoding proceeds as usual. The process is illustrated in Figure A.1.

A.3 Replacements

A.3.1 General

Replacements allow to replace part of an image file or the entire image file with different content. Four types of replacements can be distinguished:

— box replacement type: replaces a box with one or more boxes;

— app marker segment replacement type: replaces an app marker segment with one or more app marker segments (can only be used with JPEG-1 and JPEG XT);

— region of interest replacement type: replaces a region of interest in the image stream with another image;

— File replacement type: replaces the entire file.

When a JUMBF Replacement box is encountered, the associated replacement shall be applied first. Thereafter, the resulting file is decoded as usual. Except in case of a ROI replacement the image stream and ROI are first decoded individually. Subsequently the ROI is inserted in the signalled region of the decoded image. The process is illustrated in Figure A.5.

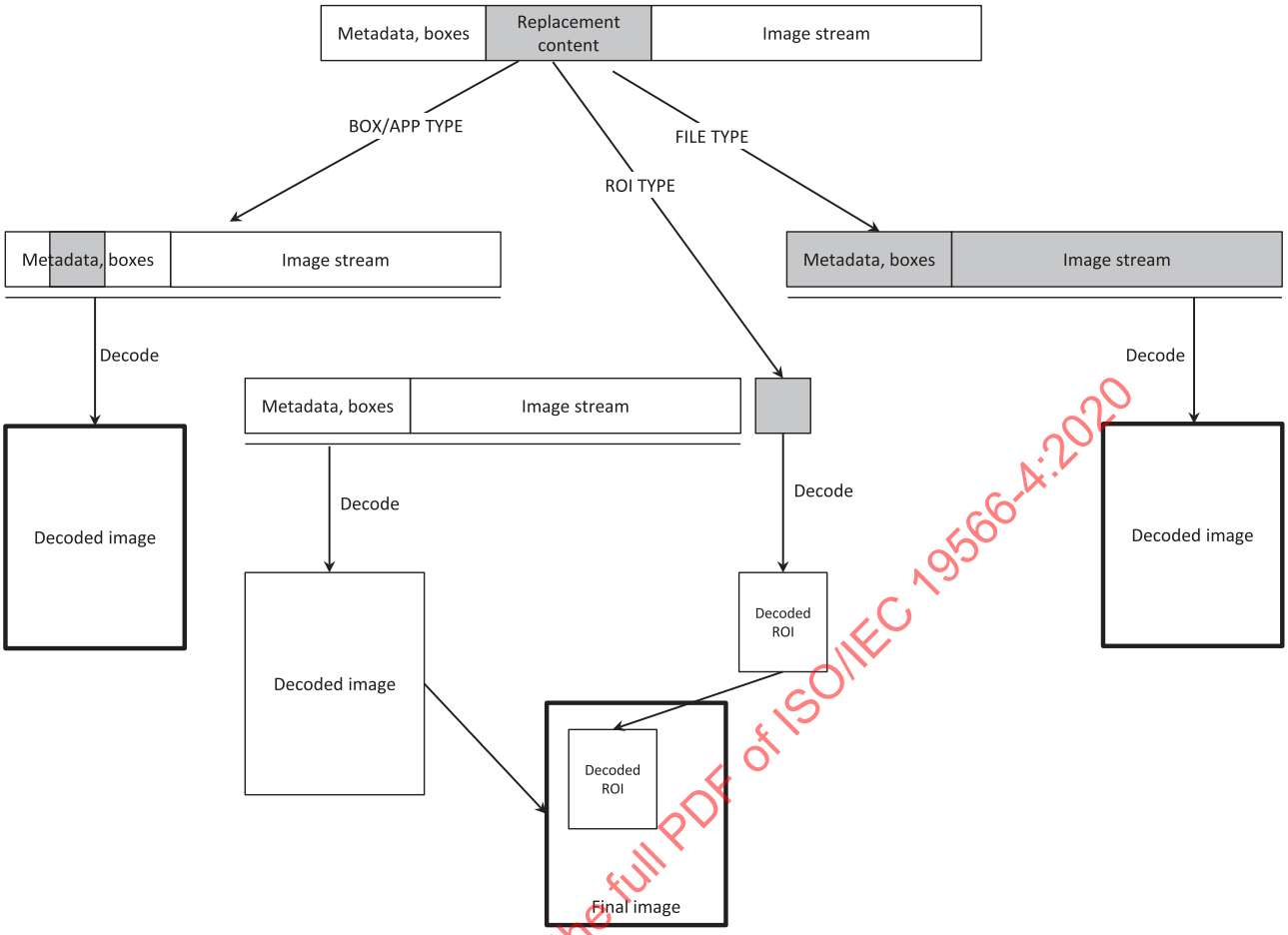


Figure A.5 — Replacement process

In case multiple replacements occur in a single file, the replacements should be applied sequentially in the order that they appear in the file.

A.3.2 JUMBF Replacement box

The JUMBF Replacement box is a JUMBF box with Content Type Replacement (UUID: DC28B95F-B68A-498E-8064-0FCA845D6B0E). The structure of the JUMBF Replacement box is given in Figure A.6.

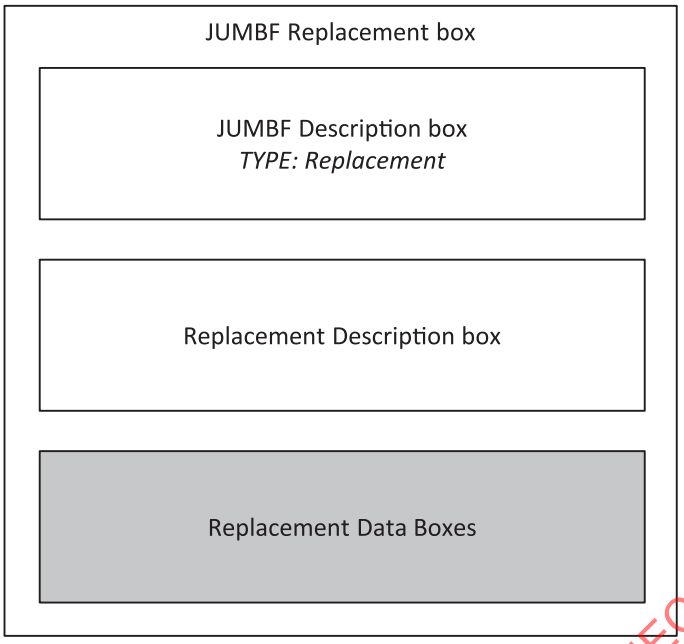


Figure A.6 — Structure of the JUMBF Replacement box

In addition to the JUMBF Description box the JUMBF Replacement box contains a Replacement Description box and one or more boxes referred to as Replacement Data Boxes. The valid box types of the Replacement Data Boxes are determined by the type.

A.3.3 Replacement Description box

The Replacement Description box signals additional information about the behaviour and type of the replacement.

The type of a Replacement Description box shall be 'psrd' (0x7073 7264). The binary structure of the Replacement Description box is illustrated in Figure A.7 and summarized in Table A.7.

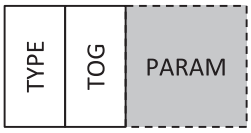


Figure A.7 — Structure of the Replacement Description box

The allowed values and meanings of the TYPE and TOG(GLES) fields are described in Table A.4 and Table A.5 respectively.

Table A.4 — TYPE specification

TYPE binary value	Meaning	Details
0000 0000	**TYPE BOX**	Box replacement type.
0000 0001	**TYPE APP**	App marker segment replacement type.
0000 0010	**TYPE ROI**	Region of interest replacement type.
0000 0011	**TYPE FILE**	File replacement type.
All other values are reserved for future use.		

Table A.5 — TOG(GLES) specification

TOG(GLES) binary value	Meaning	Details
0000 0000	**Auto apply**	**Auto apply.** Signals if the replacement should be applied by default or ignored. If auto apply is disabled, the replacement can be triggered on demand.
0000 0001	**Do not auto apply**	
All other values are reserved for future use.		

The structure of the PARAM field is determined by the replacement TYPE. Figure A.8 gives an overview of the structure of the Replacement Description box per Replacement TYPE.

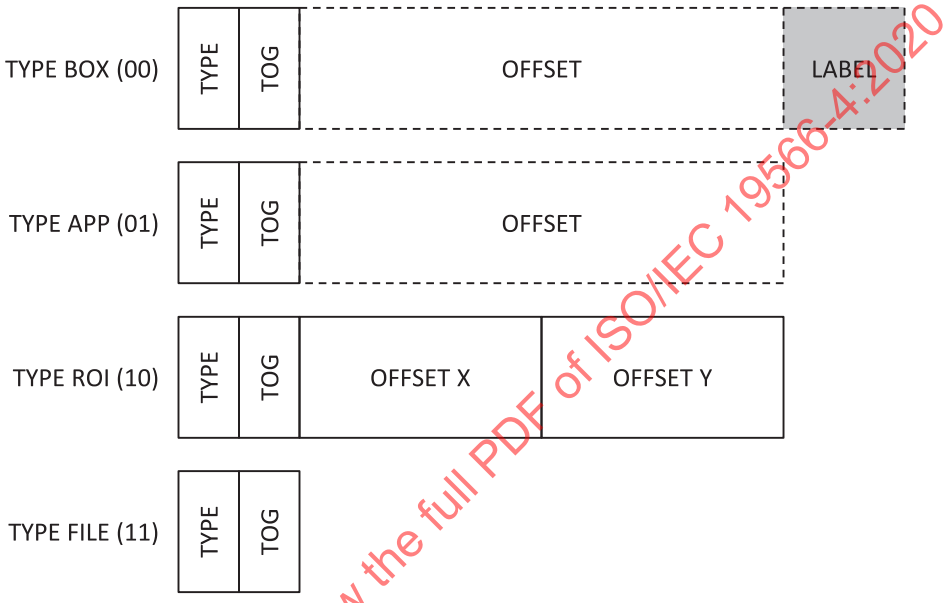


Figure A.8 — Structure of the Replacement Description box per TYPE

Table A.6 gives an overview of the field values and meanings.

Table A.6 — Description of the PARAM fields per TYPE

Replacement TYPE	Parameter	Size (bits) and type	Value and details
BOX	**OFFSET**	64; Unsigned Integer	64-bit value of the byte offset of the position of the Replacement target box counting from the beginning of the file. When set to 0, replace the successive box. When set to 2^{64}-1 (0xFFFFFFFFFFFFFFFF), replace the JUMBF box with the label specified in the LABEL field.
	LABEL	Variable; Null terminated UTF-8 string.	Label of the Replacement target JUMBF box.
APP	**OFFSET**	64; Unsigned Integer	64-bit value of the byte offset of the position of the Replacement target app marker segment counting from the beginning of the file. When set to 0, replace the successive APP marker segment.

Table A.6 *(continued)*

Replacement TYPE	Parameter	Size (bits) and type	Value and details
ROI	**OFFSET X**	32; Unsigned Integer	Vertical offset from the top in pixels.
	OFFSET Y	32; Unsigned Integer	Horizontal offset from the left in pixels.

Table A.7 — Format of the contents of the JUMBF Description box

Parameter	Size (bits)	Value
TYPE	8	See Table A.4
TOGGLES (TOG)	8	See Table A.5
PARAM	variable	See Table A.6

A.3.4 Box replacement

The referenced box is replaced with all Replacement Data Boxes. The box is either referenced by offset, JUMBF label, or position of the JUMBF Replacement box, as specified in Table A.6.

A.3.5 APP marker segment replacement

With the APP marker segment replacement type the Replacement Data boxes shall be exactly one Binary Data box. The referenced APP marker segment is replaced with the content of the Binary Data box. The APP marker segment is either referenced by offset or position of the JUMBF Replacement box if the offset is set to 0, as specified in Table A.6.

A.3.6 ROI replacement

Replaces a region of interest (ROI) in the parent image. The replacement is applied after decoding the parent image. The Replacement Data boxes shall contain exactly one Contiguous Codestream box.

NOTE In ISO/IEC 10918-1 (JPEG-1) and ISO/IEC 18477 (JPEG XT) images the replacement ROI may be a JPEG XT image with a transparency layer.

The position of the ROI is determined by the vertical and horizontal offsets x and y as illustrated in Figure A.9. The height of the ROI should be smaller than the size of the parent image minus offset x and the width smaller than width master image minus offset y.

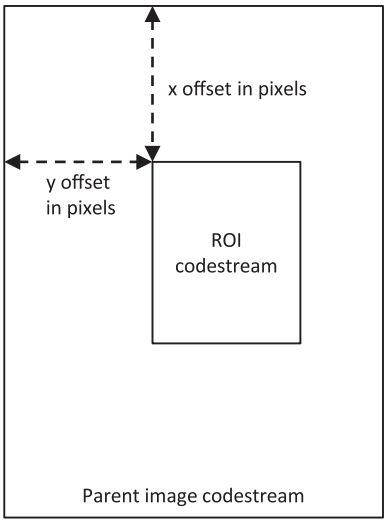


Figure A.9 — Illustration of the x and y offsets of the ROI relative to the master image

A.3.7 File replacement

Replaces the entire file. The Replacement Data boxes shall contain exactly one Contiguous Codestream box that contains the replacing image.

Annex B
(informative)

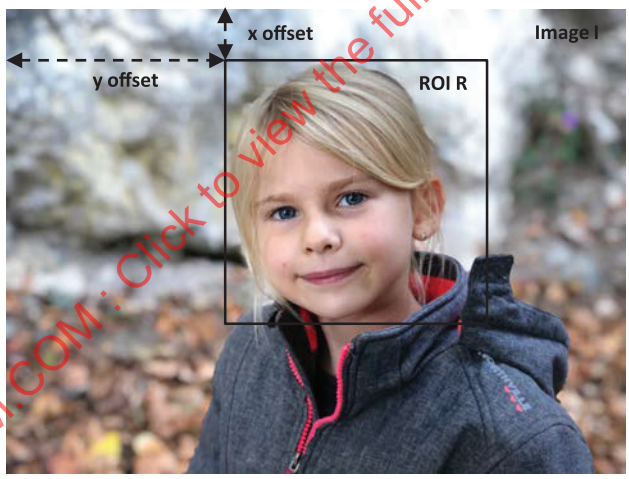
Usage examples

B.1 General

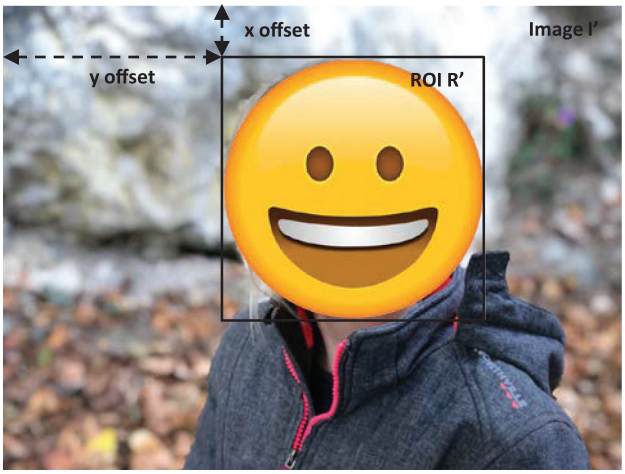
Usage examples in B.2 and B.3 illustrate how to protect part of an image using the default encryption scheme and an external encryption scheme respectively. The example in B.4 shows how to associate access rules to protected content. Finally, usage examples in B.5 and B.6 focus on image integrity. Integrity checking of images can be done by two methods: a signature- and a watermark-based method. B.5 describes examples of encapsulating and checking integrity using the signature-based method while B.6 focuses on the watermark-based method.

B.2 Protecting a region of interest in a JPEG-1 image

This usage example explains how to protect a specific region of interest in a JPEG-1 image using the JPEG privacy and security framework. The image in Figure B.1 a) contains a face that we would like to protect for privacy reasons before sharing it. Only users with appropriate rights should be able to see the actual face. Non-authorized users will see a picture where the face is not visible, as shown in Figure B.1 b).



a)



b)

Figure B.1 — Protecting a ROI in a JPEG image

Starting from the original image, we proceed through the following steps:

1. Crop out the desired region of interest of the original image (I) and store as a separate image stream (R).
2. Create an alternate version R' of the region of interest R in which the face is made invisible, for example by applying blur or adding an overlay as in Figure B.1. R' should have exactly the same dimensions as R.
3. Merge the alternate region R' into the original image at the same position of R defined by an offset X and an offset Y. Store the resulting image as I'.
4. Add a JUMBF Replacement box with replacement type ROI to the image file of I' that contains the codestream R and the X and Y offsets.
5. Wrap the JUMBF Replacement box in a JUMBF Protection box.

Decoding the resulting files proceeds as follows:

1. While scanning the image file for JUMBF boxes, the JUMBF Protection box will be identified. If the user does not have the appropriate rights to decrypt the content, then it will ignore the entire JUMBF Protection box and decode the image as usual. The user will only see the protected image I'.
2. If the user has the appropriate rights, the encrypted content is decrypted resulting in the JUMBF Replacement box which replaces the JUMBF Protection box in the file.
3. The image stream R' which is embedded in the JUMBF Replacement box is merged with the image stream I' at the position determined by offsets x and y. The result is image stream I that replaces image steam I' in the image file.
4. The resulting image file can now be decoded as usual. The user will now see the original image I.

Figure B.2 shows the structure of the image file with the protected region of interest.

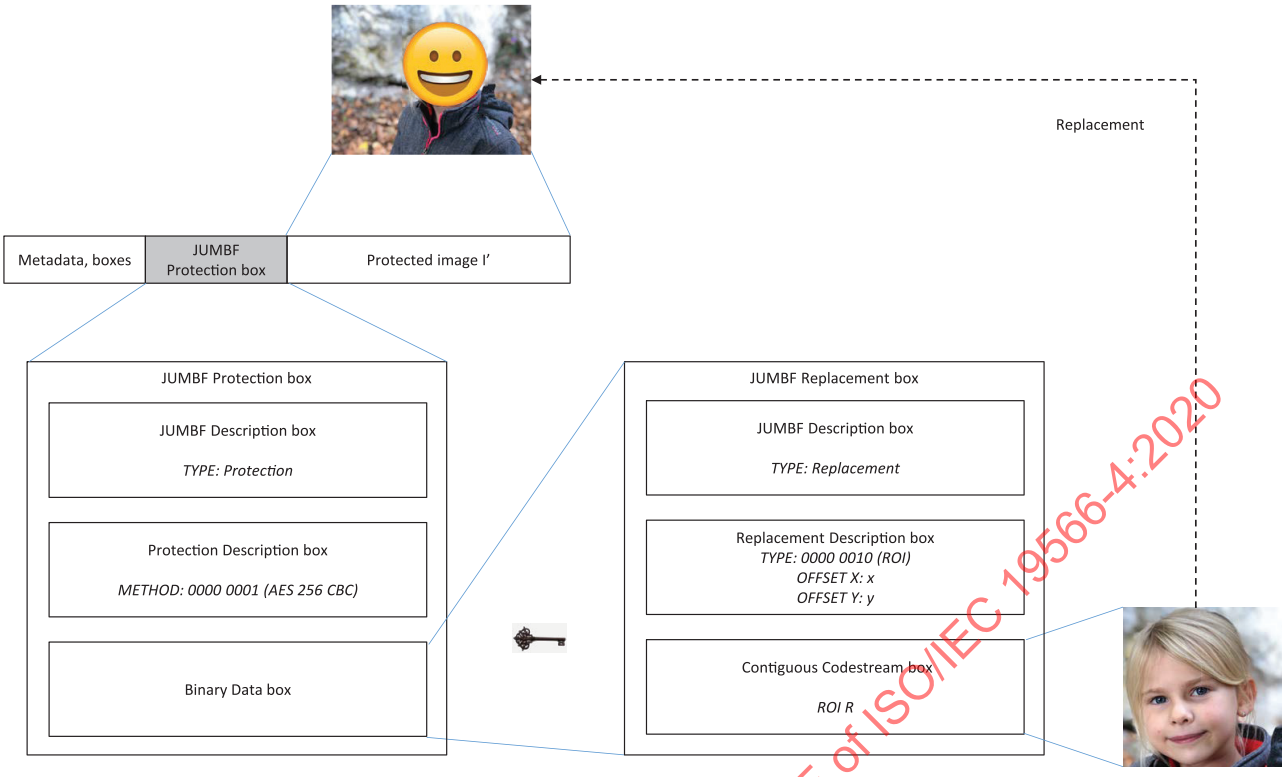


Figure B.2 — Structure of an image with a protected region of interest

B.3 Protecting with external encryption method

This example illustrates how to protect a JPEG-1 image using another encryption method than the default specified in the JPEG privacy and security framework. The example is analogue to example B.2 except that now another encryption method is used instead of the default AES 256 CBC. Therefore, the METHOD field in the Protection Description box is set to 0000 0000 and is followed by a label field that references a JUMBF box that provides more information about the used encryption. The structure is illustrated in Figure B.3. The Public Key Cryptography Standard (PKCS) has several cipher operation modes, such as ECB, CFB, OFB, CTR, CCM and GCM[3]. If the operation mode is CCM or GCM, the cipher setting has more parameters such as nonce, aad (additional authenticated data) and authentication tag[4]. In this case, the JUMBF box has the JSON type and embeds valid JSON instance that signals the additional parameters. An example JSON instance is given in Table B.1.

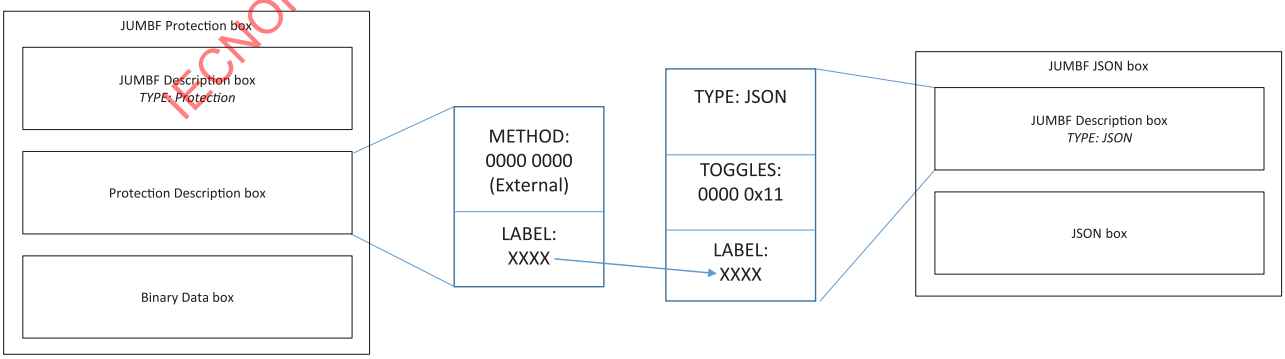


Figure B.3 — JUMBF Protection box referring to JSON Content Type JUMBF box

Table B.1 — Sample JSON for additional cipher parameters (AES256-GCM)

```
{
            "jpeg_security": {
                "type": "protection",
                "cipher": {
                    "method": "AES256-GCM",
                    "nonce": "BdZbHABY/sytDTUB",
                    "aad": "ZmFzb28uY29t",
                    "tag": "1dsCuZ5XuanojwM/p6EoCA=="
                }
            }
    }
```

B.4 Associating access rules to protected content

B.4.1 Usage example description

As indicated in A.2.3, the access rules associated to an image, if any, are stored in a referenced JUMBF Box.

Access rules may be expressed using eXtensible Access Control Markup Language (XACML)[1], so in this case the JUMBF box must be of XML Content type.

Figure B.4 shows how the JUMBF Protection box can reference the JUMBF XML box containing the corresponding access rules. In this example, AES 256 CBC encryption method is used, as indicated in the METHOD field with the binary code 0001 0001.

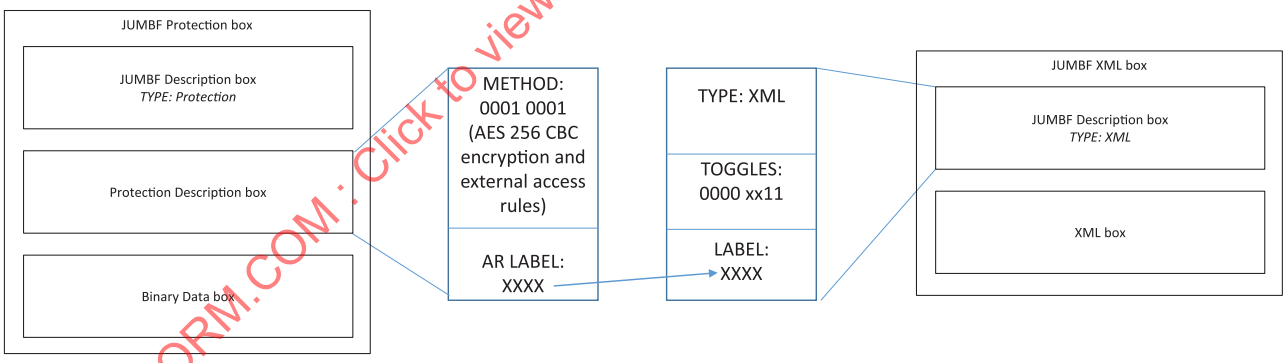


Figure B.4 — JUMBF Protection box referring to XML Content Type JUMBF Box

Table B.2 shows a complete XACML policy, containing several access rules, which is the content of the JUMBF XML box. The first rule inside the policy defines the conditions to indicate that any user can view the image urn:mimage:Desert.jpg before 2020-01-01. The second one denies any other operation over the image.

Table B.2 — XML Content Type JUMBF box for a XACML policy containing several access rules

```
<Policy xmlns="urn:oasis:names:tc:XACML:3.0:core:schema:wd-17"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     PolicyId="urn:isdcm:policyid:1"
     RuleCombiningAlgId="urn:oasis:names:tc:XACML:1.0:rule-combining-algorithm:first-
applicable"
     Version="1.0"
     xsi:schemaLocation="urn:oasis:names:tc:XACML:3.0:core:schema:wd-17
     http://docs.oasis-open.org/XACML/3.0/XACML-core-v3-schema-wd-17.xsd">
     <Description>Desert.jpg</Description>
     <Rule Effect="Permit" RuleId="urn:oasis:names:tc:XACML:2.0:ejemplo:Desert">
       <Description>
         Any user can view urn:mimage:Desert.jpg before the end of the year
       </Description>
       <Target>
          <AnyOf>
              <AllOf>
                  <!-- Which resource -->
                  <Match
                    MatchId="urn:oasis:names:tc:XACML:1.0:function:regexp-string-match">
                    <AttributeValue
                      DataType="http://www.w3.org/2001/XMLSchema#string">
                        urn:mimage:Desert.jpg
                    </AttributeValue>
                    <AttributeDesignator
```

Table B.2 *(continued)*

```
                    AttributeId="urn:oasis:names:tc:XACML:1.0:resource:resource-id"
                    Category="urn:oasis:names:tc:XACML:3.0:attribute-category:resource"
                    DataType="http://www.w3.org/2001/XMLSchema#string"
                    MustBePresent="false"/>
                </Match>

                <!-- Which action -->
                <Match MatchId="urn:oasis:names:tc:XACML:1.0:function:string-equal">
                  <AttributeValue
                    DataType="http://www.w3.org/2001/XMLSchema#string">
                      View
                  </AttributeValue>
                  <AttributeDesignator
                    AttributeId="urn:oasis:names:tc:XACML:1.0:action:action-id"
                    Category="urn:oasis:names:tc:XACML:3.0:attribute-category:action"
                    DataType="http://www.w3.org/2001/XMLSchema#string"
                    MustBePresent="false"/>
                </Match>
            </AllOf>
        </AnyOf>
      </Target>
      <Condition>
        <Apply FunctionId="urn:oasis:names:tc:XACML:1.0:function:date-less-than-or-equal">
            <Apply FunctionId="urn:oasis:names:tc:XACML:1.0:function:date-one-and-only">
                <AttributeDesignator AttributeId="accessDate"
                  Category="urn:oasis:names:tc:XACML:3.0:date"
                  DataType="http://www.w3.org/2001/XMLSchema#date"
MustBePresent="false"/>
            </Apply>
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#date">
                2020-01-01
            </AttributeValue>
        </Apply>
      </Condition>
    </Rule>
    <Rule RuleId="urn:oasis:names:tc:XACML:2.0:FinalRule" Effect="Deny"/>
</Policy>
```


B.4.2 Encryption and access rules addition process

Figure B.5 shows how an image is encrypted and the access rules governing its usage are also added to the resulting file.

Starting from the original image, metadata is transferred to the resulting image. Then, the image stream is encrypted and the result of the encryption is stored in the Binary Data box inside the Protection box. Afterwards, the access rules and encryption information are added to the Protection box. The information related to access rules is referenced by means of a label. Finally, a new XML Content type JUMBF box is added, which contains the XACML access rules that govern the usage of the protected image file.

Figure B.6 shows the new JUMBF boxes resulting from the encryption and access rules addition process.

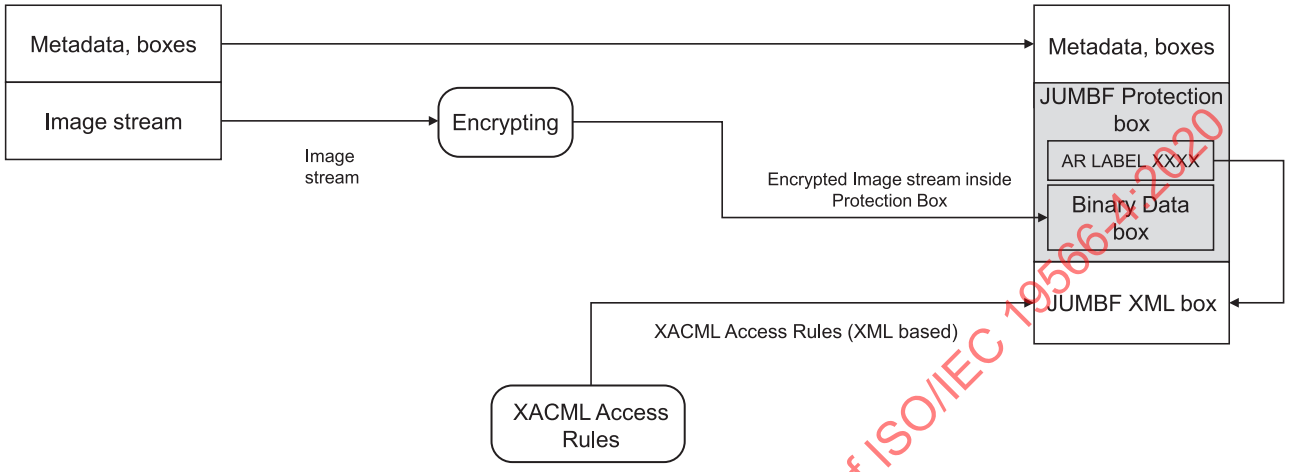


Figure B.5 — Encrypting and adding XACML access rules to an image

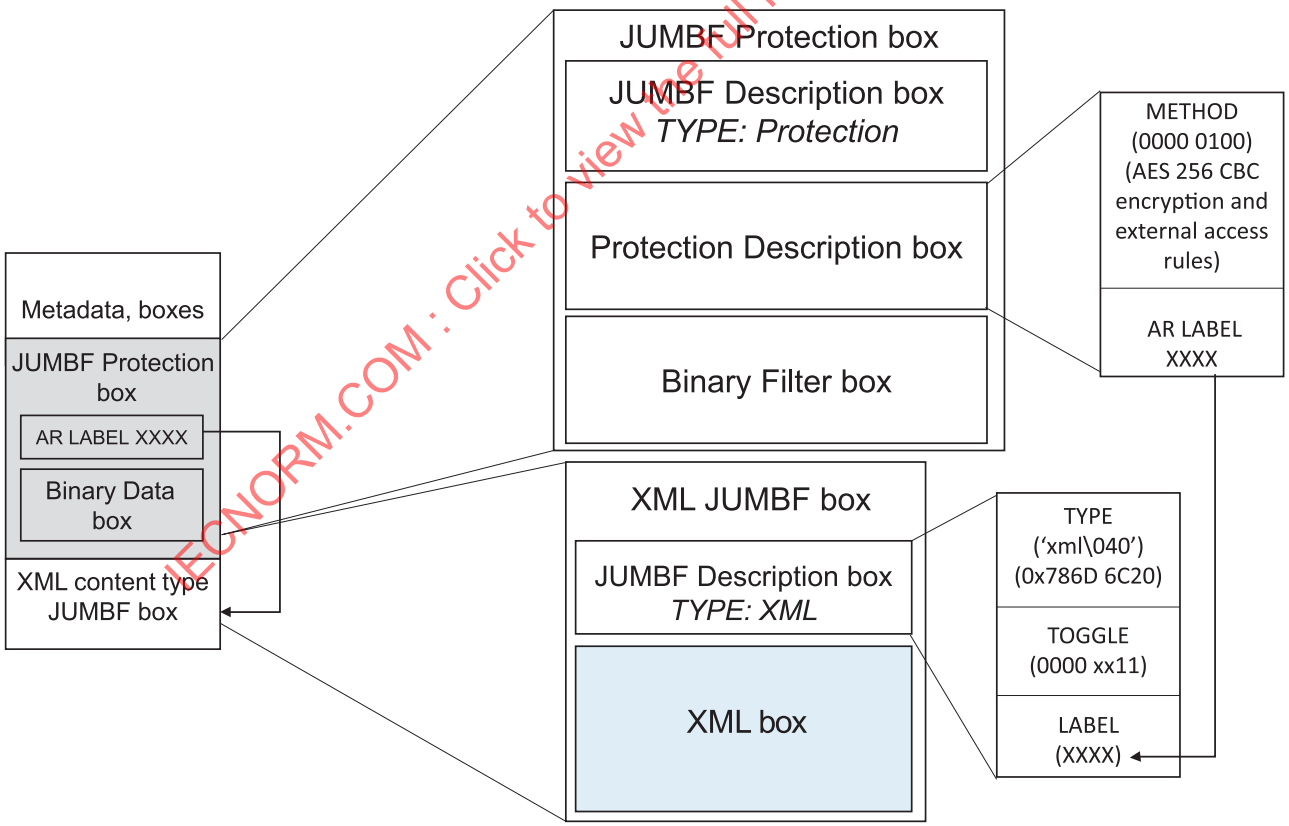


Figure B.6 — Generated JUMBF boxes by encryption and access rules process

B.4.3 Authorization and decryption process

Figure B.7 shows the authorization and decryption (if any) process when access rules are present. In this case, first, the label referencing the XML content type JUMBF box containing the access rules (LABEL XXXX inside the JUMBF protection box) can be obtained from the JUMBF protection box. Then, access rules can be extracted from the XML content type JUMBF box. The access rules are given to the authorization service, which also needs an authorization request (as specified in the XACML standard[1]). Then, if the authorization is granted, the decryption process obtains the encryption information and the encrypted image from the JUMBF Protection box. After that, the decryption process performs the decryption of the image, generating a result file with the corresponding metadata and the decrypted image stream. If the operation is not authorized according to the access rules, the resulting image contains only the image information which was not encrypted in the original image.

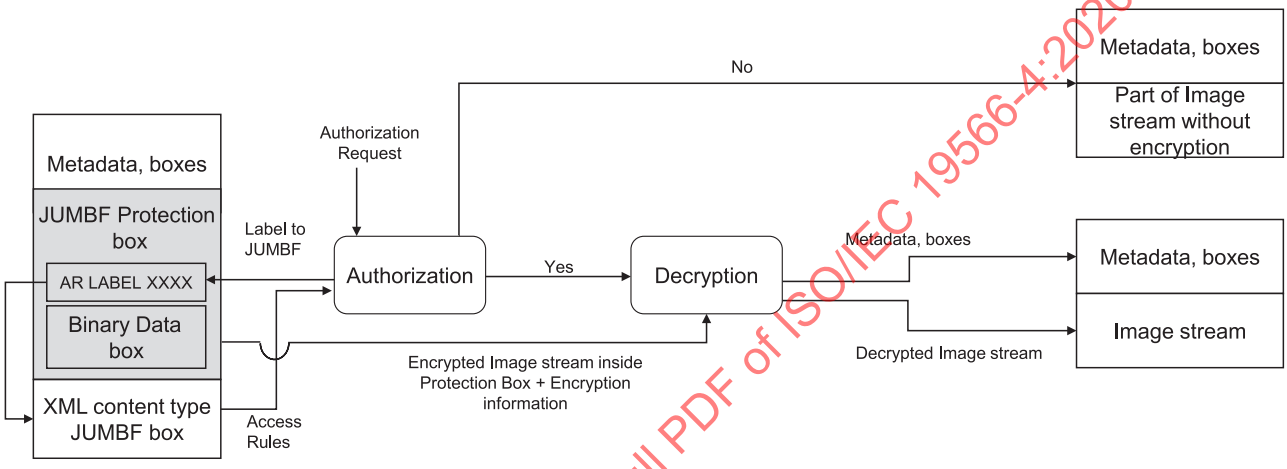


Figure B.7 — Authorizing access and decrypting an image

B.5 Integrity checking using a signature-based method

B.5.1 Including a signature to enable integrity checking functionality

Figure B.8 provides an example of encapsulating the functionality of the integrity checking using a signature-based method. This example checks if an image stream and/or a selected part of metadata are modified or not. The owner's key is used for this encapsulating procedure. If this key is mismatched when the integrity is checked, the result of the check is always that the checked image is the modified one.

SHA-256 hashing is used for signature generating process in Figure B.8. The input to this process is a set of the image stream, the selected metadata, and the owner's key. The resulting signature and index of a selected part of metadata is saved as an XML content type JUMBF box.

Figure B.9 shows the type of a generated JUMBF box by this encapsulating procedure. Table B.3 shows an XML schema that defines types, attributes, and elements for sample XML documents used in B.5.1 and B.6.1. Table B.4 shows a sample XML document for the signature and the index of a selected part of metadata following XML schema defined in Table B.3.

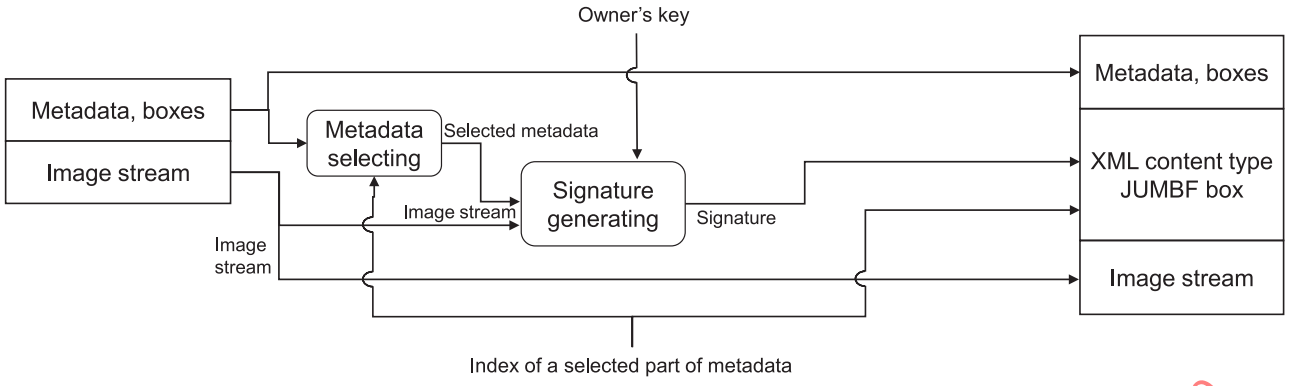


Figure B.8 — Example of encapsulating procedure using a signature-based method

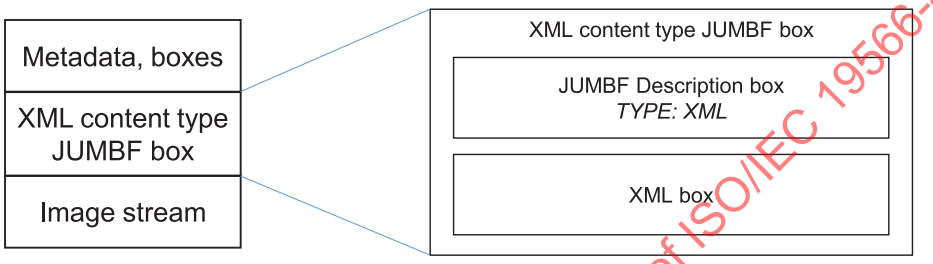


Figure B.9 — Generated JUMBF box by encapsulating procedure