INTERNATIONAL STANDARD

ISO/IEC 15444-8

Second edition
2023-10

Information technology — JPEG 2000 image coding system —

Part 8: Secure JPEG 2000

Technologies de l'information — Système de codage d'images JPEG 2000 —

Partie 8: JPEG 2000 sécurisé

Reference number
ISO/IEC 15444-8:2023(E)

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

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.

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take 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 and IEC 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 and https://patents.iec.ch. ISO and IEC 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. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by ITU-T (as ITU-T Rec T.807) and drafted in accordance with its editorial rules, in collaboration with Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This second edition cancels and replaces the first edition (ISO/IEC 15444-8:2007), which has been technically revised. It also incorporates the Amendment ISO/IEC 15444-8:2007/Amd 1:2008.

A list of all parts in the ISO/IEC 15444 series can be found on the ISO and IEC websites.

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 and www.iec.ch/national-committees.

INTERNATIONAL STANDARD ISO/IEC 15444-8
RECOMMENDATION ITU-T T.807

Information technology – JPEG 2000 image coding system: Secure JPEG 2000

Summary

Rec. ITU-T T.807 | ISO/IEC 15444-8 provides a syntax that allows security services to be applied to JPEG 2000 coded image data. Security services include confidentiality, integrity verification, source authentication, conditional access, secure scalable streaming and secure transcoding. The syntax allows these security services to be applied to coded and uncoded image data in part or in its entirety. This maintains the inherent features of JPEG 2000 such as scalability and access to various spatial areas, resolution levels, colour components, and quality layers, while providing security services on these elements.

This second edition:

1) consolidates all solved/outstanding amendments and corrigenda published since the first edition;
2) removes the clause on Registration Authority (first edition's clause 7);
3) removes the annex of patent statements (first edition's Annex D).

This Recommendation | International Standard was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*, in collaboration with ITU-T SG16. The identical text is published in ISO/IEC as ISO/IEC 15444-8.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T T.807	2006-05-29	16	11.1002/1000/8830
1.1	ITU-T T.807 (2006) Amd. 1	2008-03-15	16	11.1002/1000/9364
2.0	ITU-T T.807 (V2)	2023-02-13	16	11.1002/1000/15208

Keywords

Image security, jpsec, JPEG 2000, security, still image.

* To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents/software copyrights, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the appropriate ITU-T databases available via the ITU-T website at http://www.itu.int/ITU-T/ipr/.

CONTENTS

FIGURES

Page

Page

Page

TABLES

Page

Page

Page

Page

Introduction

In the "Digital Age", the Internet provides many new opportunities for right-holders regarding the electronic distribution of their work (books, videos, music, images, etc.).

At the same time, new information technology radically simplifies the access of content for the user. This goes hand in hand with the all-pervasive problem of pirated digital copies – with the same quality as the originals – and "file-sharing" in peer-to-peer networks, which gives rise to continued complaints about great losses by the content industry.

World Intellectual Property Organization (WIPO) and its Member countries (170) have an important role to play in assuring that copyright, and the cultural and intellectual expression it fosters, remains well protected in the 21st century. The new Digital economy and the creative people in every country of the world depend on it. Also in December 1996, WIPO Copyright Treaty (WCT) has been promulgated with two important articles (11 and 12) about technological measures and obligations concerning Right Management Information:

Article 11
Obligations concerning
Technological Measures

Contracting Parties shall provide adequate legal protection and effective legal remedies against the circumvention of effective technological measures that are used by authors in connection with the exercise of their rights under this Treaty or the Berne Convention and that restrict acts, in respect of their works, which are not authorized by the authors concerned or permitted by law.

Article 12
Obligations concerning Rights
Management Information

(1) Contracting Parties shall provide adequate and effective legal remedies against any person knowingly performing any of the following acts knowing, or with respect to civil remedies having reasonable grounds to know, that it will induce, enable, facilitate or conceal an infringement of any right covered by this Treaty or the Berne Convention:

(i) to remove or alter any electronic rights management information without authority;

(ii) to distribute, import for distribution, broadcast or communicate to the public, without authority, works or copies of works knowing that electronic rights management information has been removed or altered without authority.

(2) As used in this Article, "rights management information" means information which identifies the work, the author of the work, the owner of any right in the work, or information about the terms and conditions of use of the work, and any numbers or codes that represent such information, when any of these items of information is attached to a copy of a work or appears in connection with the communication of a work to the public.

This treaty provides a solid foundation to protect Intellectual Property. As of 2004, about 50 countries ratified this important treaty. Therefore, it is expected that tools and protective methods that are recommended in JPEG 2000 needs ensure the security of transaction, protection of content (IPR), and protection of technologies.

Security issues, such as authentication, data integrity, protection of copyright and Intellectual Property, privacy, conditional access, confidentiality, transaction tracing, to mention a few, are among important features in many imaging applications targeted by JPEG 2000.

The technological means of protecting digital content are described and can be achieved in many ways such as digital watermarking, digital signature, encryption, metadata, authentication, and integrity checking.

This Recommendation | International Standard intends to provide tools and solutions in terms of specifications that allow applications to generate, consume, and exchange Secure JPEG 2000 codestreams. This is referred to as **JPSEC**.

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – JPEG 2000 image coding system: Secure JPEG 2000

1 Scope

This Recommendation | International Standard specifies the framework, concepts, and methodology for securing JPEG 2000 codestreams. The scope of this Recommendation | International Standard is to define:

1) a normative codestream syntax containing information for interpreting secure image data;
2) informative examples of JPSEC tools in typical use cases;
3) informative guidelines on how to implement security services and related metadata.

The scope of this Recommendation | International Standard is not to describe specific secure imaging applications or to limit secure imaging to specific techniques, but to create a framework that enables future extensions as secure imaging techniques evolve.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies. At the time of publication, the editions indicated in dated references were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

- Recommendation ITU-T T.800 | ISO/IEC 15444-1, *Information technology – JPEG 2000 image coding system: Core coding system*.
- Recommendation ITU-T T.801 | ISO/IEC 15444-2, *Information technology – JPEG 2000 image coding system: Extensions*.
- Recommendation ITU-T T.808 | ISO/IEC 15444-9, *Information technology – JPEG 2000 image coding system: Interactivity tools, APIs and protocols.*
- Recommendation ITU-T T.814 | ISO/IEC 15444-15, *Information technology – JPEG 2000 image coding system: High-Throughput JPEG 2000.*
- Recommendation ITU-T X.509 | ISO/IEC 9594-8, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*.
- ISO/IEC 14496-12, *Information technology* – Coding of audio-visual objects – *Part 12: ISO base media file format*.

3 Definitions

For the purposes of this document, the definitions contained in Rec. ITU-T T.800 | ISO/IEC 15444-1, Rec. ITU-T T.801 | ISO/IEC 15444-2, Rec. ITU-T T.808 | ISO/IEC 15444-9, Rec. ITU-T T.814 | ISO/IEC 15444-15, and ISO/IEC 14496-12, and the following apply.

3.1 **access control**: Prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner.

3.2 **adaptive-format decoder**: Process to decode a codestream which is not fully compliant with the normative part of the coding standard.

NOTE – It reconstructs the media (possibly with low quality or resolution) even if the codestream has missing packets or inconsistent packet headers. For example, an adaptive-format decoder is able to understand a simply transcoded codestream, such as the one that has its highest resolution packets removed.

3.3 **adaptor/transcoder**: process to transform media data to lower scalability level, like lower resolution or lower quality or bit-rate, by removing portions of the file.

NOTE – The adaptor/transcoder can transform media data based on the information specified in this Recommendation | International Standard. An adaptor/transcoder updates byte offset values in file format parameters that are impacted by the process.

3.3.1 authentication adaptor/transcoder: Process to remove data that is not verifiable with the available media and authentication data.

NOTE – For example, in a streaming system, some media packets can be lost during transmission. A file format receiver can reconstruct the received data to the best of its ability based on the available data. Then, an authentication adaptor/transcoder can determine which data can be verified, and then remove the packets that are not verified. The resulting file only contains the decodable, verified data.

3.3.2 JPEG 2000-aware adaptor/transcoder: Process that combines one or more scalable composed ESs to form a fully decodable media codestream.

3.3.3 simple adaptor/transcoder: Process to transform data based on information specified by this Recommendation | International Standard.

NOTE – A simple adaptor/transcoder might not be capable of generating media headers or modifying packet indices. It simply retrieves data pointed by pointer structure and removes the wrappers, and the resulting codestream can be decoded by adaptive-format decoder, which can cope with missing packets and inconsistent headers.

3.4 authentication: Process of verifying an identity claimed by or for a system entity.

3.4.1 source authentication: Verification that a source entity (say, user/party) is in fact the claimed source entity.

3.5 ByteData: Structure used to wrap a data segment which is physically located in a composed ES.

NOTE – It is contained inside a container structure.

3.6 confidentiality: Property that information is not made available or disclosed to unauthorized individuals, entities or processes.

3.7 container: Container structure is used to wrap a sample in a composed ES.

NOTE – It might contain any number of ByteData or pointer structures, but is not allowed to contain another container structure.

3.8 data splitting: Method to protect sensitive data from unauthorized access by encrypting the data and storing different portions of the file on different servers.

NOTE – When split data is accessed the parts are retrieved, combined and decrypted. An unauthorized person would need to know the locations of the servers containing the parts, be able to get access to each server, know what data to combine, and how to decrypt it.

3.9 decryption, deciphering: Inverse transformation of the encryption.

3.10 digital signature: Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery, e.g., by the recipient.

3.11 elementary stream (ES): Contains a sequence of samples, where each sample could be a video frame or a contiguous section of audio data.

NOTE – A sample in ES contains media data, ByteData structure, pointer structure, container structure, or any mixture of these.

3.11.1 self-contained ES: ES that contains only media data, whose format is not defined in this Recommendation | International Standard.

NOTE – The self-contained ES could be stored in MDAT box co-located with the file format specified in this Recommendation | International Standard, or be stored in a separate file whose format is not specified by this Recommendation | International Standard.

3.11.2 composed ES: ES that can contain a mixture of ByteData, pointer and container structures, that is, its samples are composed with data from other elementary streams.

NOTE – A composed ES can either copy (using ByteData structure) or reference (using pointer) data from other ESs.

3.11.3 scalable composed ES: ES made up of samples that might not be decodable by themselves.

NOTE – A scalable composed ES might need to be combined with other scalable composed ESs to form a fully decodable codestream. Scalable composed ES is designed to support scalability, i.e., to make media data "thinable". For example, for a Motion JPEG 2000 codestream where each picture has three layers, it can be divided into 3 scalable composed ESs: the first one consists of all layer 0 data, the second one consists of all layer 1 data and the third one consists of all layer 2 data.

3.11.4 decodable composed ES: ES made up of samples that are decodable by themselves.

NOTE – It is designed for simple adaptation where the adaptor just needs to retrieve data pointed by pointer structure and remove the wrapper to form a fully scalable codestream. For example, for a Motion JPEG 2000 codestream where each picture has three layers, it can form 3 decodable composed ESs: the first one consists of layer 0 data, the second one consists of layer 0 and layer 1 data and the third one consists of layer 0, 1 and 2 data.

3.12 encryption: Reversible transformation of data by a cryptographic algorithm to produce ciphertext, i.e., to hide the information content of the data.

NOTE – An alternative term for an encryption algorithm is cipher.

3.13 **hash function**: Function which maps strings of bits to fixed-length strings of bits.

NOTE 1 – This function satisfies the following two properties:

- For a given output, it is computationally infeasible to find an input which maps to this output.
- For a given input, it is computationally infeasible to find a second input which maps to the same output.

NOTE 2 – Computational feasibility depends on the user's specific security requirements and environment.

3.14 **integrity**: Property of being able to safeguard the accuracy and the completeness of assets.

3.14.1 **image data integrity**: Property that data has not been altered or destroyed in an unauthorized manner.

3.14.2 **image content integrity**: Assurance the image content has not been modified by unauthorized parties in such a way that its perceptual meaning is changed.

NOTE – It allows the content-preserving operations to be performed on the image without triggering the integrity alarm.

3.15 **JPSEC application**: Software or hardware process that is capable of consuming JPSEC codestreams by interpreting the JPSEC syntax in order to provide the specified security services.

NOTE – A JPSEC application makes use of one or several JPSEC tools.

EXAMPLE – A JPSEC application would be able to read encrypted JPSEC codestreams, decrypt them when provided with the appropriate key and render the JPEG 2000 original clear-text image data.

3.16 **JPSEC codestream**: Sequence of bits resulting from coding and securing an image using JPEG 2000 coding and JPSEC security tools.

3.16.1 **JPSEC creator**: Entity who creates a JPSEC codestream from an image, a JPEG 2000 codestream, or another JPSEC codestream in order to provide some JPSEC services.

3.16.2 **JPSEC consumer**: Entity who receives a JPSEC codestream and renders a JPSEC service based on the codestream.

3.17 **JPSEC service**: Service that provides security for consumption of JPEG 2000 images. The service counters security attacks and makes use of one or several JPSEC tools.

3.18 **JPSEC tool**: Hardware or software process that uses security techniques to implement a security service.

3.18.1 **JPSEC normative tool**: JPSEC tool that uses predefined tool templates for decryption, authentication, or hashing specified by the normative part of this Recommendation | International Standard.

3.18.2 **JPSEC non-normative tool**: JPSEC tool that is specified by an identification number given by a user-defined application.

NOTE – Examples of JPSEC non-normative tools are describe in Annex B.

3.18.3 **JPSEC RA tool**: JPSEC non-normative tool that definition is registered by JPSEC registration authority (RA). (Deprecated)

3.18.4 **JPSEC user-defined tool**: JPSEC non-normative tool that is defined by a user-defined application.

3.19 **JPSEC tool description**: Description of the parameters used by the JPSEC tool.

NOTE – JPSEC tool description does not describe the algorithm or method used. A JPSEC tool description consists of two parts: the parameter list and its values. In the case of JPSEC normative tools, the parameter list is given by the standard. In the case of JPSEC non-normative tools, the parameter list can be given by a user-defined application. In both cases, the parameter values are specified in the SEC and INSEC marker segments.

3.20 **key**: Sequence of symbols that controls the operations of encipherment and decipherment.

3.20.1 **asymmetric key pair**: Pair of related keys where the private key defines the private transformation, and the public key defines the public transformation.

3.20.1.1 **private key**: Key of an entity's asymmetric key pair which is intended to remain private.

3.20.1.2 **public key**: Key of an entity's asymmetric key pair which can be made public.

3.21 **key generation, key generating function**: Function which takes as input a number of parameters, at least one of which is secret, and which gives as output keys appropriate for the intended algorithm and application.

NOTE – The function has the property that it is computationally infeasible to deduce the output without prior knowledge of the secret input.

3.22 **key management**: Generation, storage, distribution, deletion, archiving and application of keys in accordance with a security policy.

3.23 **marker emulation**: Cipher text resulting from the encryption process that contains a JPEG start code.

3.24 **message authentication code (MAC)**: String of bits which is the output of a MAC algorithm.

3.25 **non-repudiation**: Binding of an entity to a transaction in which it participates, so that the transaction cannot later be repudiated (denied).

NOTE – That is, the receiver of a transaction is able to demonstrate to a neutral third party that the claimed sender did indeed send the transaction.

3.26 **normal decoder**: Standard decoder is a process to decode a codestream that is fully compliant with the normative part of coding standard. Its behaviour is not defined if it tries to decode a non-compliant codestream.

3.27 **packet**: Part of the JPEG 2000 Core coding system bit stream comprising a packet header and the compressed image data from one layer of the precinct of one resolution of one tile-component.

NOTE – This is different from the term "packet" used in data transmission through network.

3.28 **pointer**: Pointer structure is used to reference a data segment in another ES. It is contained inside a container structure.

3.29 **protection**: Process to secure content.

3.29.1 **protection template**: Template or list of parameter fields necessary for the operation of a protection method.

3.29.2 **protection method**: Method used to create or consume protected content such as encryption, decryption, authentication, and integrity checking.

3.30 **security**: All aspects related to defining, achieving, and maintaining confidentiality, integrity, availability, accountability, authenticity, and reliability.

NOTE – A product, system, or service is considered to be secure to the extent that its users can rely on that it functions (or will function) in the intended way. This is usually considered in the context of an assessment of actual or perceived threats.

3.31 **signalling syntax**: Specification of the format of the JPSEC codestream that contains all the required information for consuming secure JPEG 2000 images.

3.32 **transcoding**: Operation of taking an input compressed codestream and adapting or converting it to produce an output compressed codestream that has some desired property.

EXAMPLE – The output compressed codestream can represent an image with a lower spatial resolution or lower bit rate than the input compressed codestream.

3.32.1 **secure transcoding**: Operation of performing transcoding, or adaptation, of a protected input compressed content, without unprotecting the content.

NOTE – The term secure transcoding is used, as opposed to transcoding, to stress that the transcoding operation is performed without compromising security. Secure transcoding can also be referred to as performing transcoding in the encrypted domain.

3.33 **watermark**: Signal imperceptibly added to the cover-signal in order to convey hidden data.

3.33.1 **watermarking**: Process that imperceptibly inserts data representing some information into multimedia.

NOTE – The multimedia data in one of the following two ways:

- The lossy way which means the exact cover-signal will never be able to be recovered once the watermark is embedded.
- The lossless way which means the exact cover-signal could be recovered after watermark extraction.

3.34 **4CC Code**: 32-bit identifier, normally 4 printable characters.

NOTE – This 4CC code can be used to indicate the file type, the type of file format box, type of a file format track, type of a file format sample description and type of file format track reference.

4 Symbols and abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply.

BAS	Byte Aligned Segment
FBAS	Field Byte Aligned Segment
G	Granularity
GL	Granularity Level
ICV	Integrity Check Value
INSEC	In-codestream security marker
IP	Intellectual Property related to technology
IPR	Intellectual Property Rights related to content

JPSEC	Secure JPEG 2000
KT	Key Template
LSB	Least Significant Bit
MAC	Message Authentication Code
MSB	Most Significant Bit
PD	Processing Domain
PKI	Public Key Infrastructure
PO	Processing Order
RBAS	Range Byte Aligned Segment
SEC	Security marker
T	Template
V	Values
VL	Value List
ZOI	Zone of Influence

5 JPSEC syntax

5.1 JPSEC framework overview

JPSEC defines a framework for the securing of JPEG 2000 coded data. The core of this Recommendation | International Standard is the specification of the syntax of the secure JPEG 2000 image, the *JPSEC codestream*. The syntax is targeted toward JPEG 2000 coded data and allows for protection of the entire codestream or of parts of the codestream. In all cases the protected data (i.e., the JPSEC codestream) shall follow the normative syntax defined in this Recommendation | International Standard.

To the JPSEC codestream are associated a number of *JPSEC security services* including confidentiality and authentication of origin and of content.

The *signalling syntax* specifies:

- what security services are associated with the image data;
- which *JPSEC tools* are required to deliver the corresponding services;
- how the JPSEC tools are applied;
- which parts of the image data are protected.

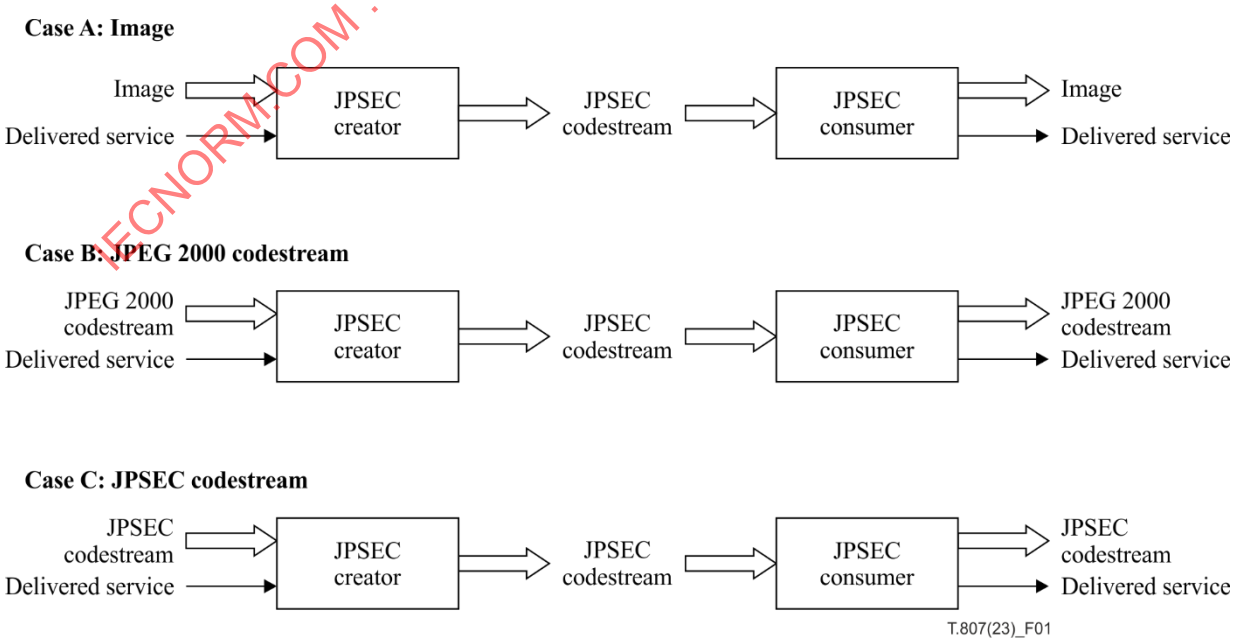


Figure 1 – Overview of the conceptual steps in JPSEC framework

The syntax of the JPSEC codestream is normative. The purpose is to allow JPSEC applications to consume JPSEC codestreams in an interoperable way (see Figure 1). The JPSEC consumer application interprets the JPSEC codestream, identifies and applies the signalled JPSEC tools, delivers the corresponding security services and then passes on the output JPEG 2000 codestream or image for subsequent processing, for example by an image viewer.

As shown in case C of Figure , the JPSEC codestream can be created from another JPSEC codestream. This can arise when multiple JPSEC tools are applied to the same content, but at different times or by different entities. When this occurs, the ordering in which the JPSEC tools are applied during the creation and consumption operations can be significant.

The signalling syntax identifies tools that are used by a JPSEC consumer. Tools are either defined by the normative part of the standard, or by private tools. The normatively defined tools support confidentiality (through encryption tools), and authentication of the source and of the content. They allow for the highest type of interoperability since independent implementations of the consuming process are able to process the same JPSEC codestream and render the corresponding services with the same behaviour.

The way in which the JPSEC codestream is created is out of scope of this Recommendation | International Standard. To be compliant, JPSEC creators shall generate JPSEC codestreams that include the appropriate JPSEC signalling. JPSEC codestreams can be created in a number of ways. For example, a JPSEC tool can be applied to image pixels or it can be applied on wavelet coefficients, or on quantized coefficients, or on packets.

A consumer can implement one or more JPSEC tools. For example, it could be capable of performing decryption using AES block cipher in ECB mode and signature verification using SHA-128 hash and an RSA public key. With these capabilities, it would be capable of performing the security services of confidentiality and authentication.

In the JPSEC framework, JPSEC tools are specified by templates, defined privately. JPSEC tools specified by the templates have unique processing behaviour and therefore do not require unique identification. Those tools are associated with a unique identification number provided by the common registry.

The guideline and use cases of JPSEC are described in Annex A, and the file format security is described in Annex D. The interoperability of JPEG 2000 family, e.g., JPX, JPIP and JPWL, are described in Annex C.

5.2 JPSEC security services

The objective in this clause is to list and to explain the functionalities that are included in the scope of this Recommendation | International Standard.

JPSEC tools are used to implement security functions. JPSEC is an open framework, which means that it is extensible in the future. Currently it focuses on the following aspects:

– Confidentiality via encryption and selective encryption

  A JPSEC file can support a transformation of the (image and/or metadata) data (plaintext) into a form (cipher text) that conceals the data's original meaning. Selective encryption means that not the entire image and/or metadata but only parts of the image and/or metadata can be encrypted.

– Integrity verification

  A JPSEC file can support means of detecting manipulations to the image and/or metadata and thereby verify their integrity. There are two classes of integrity verification:

  1) Image data integrity verification where even only one bit of image data in error results in verification failure (i.e., the verification returns "no integrity"). This verification is also often referred to as fragile image (integrity) verification.
  2) Image content integrity verification where even some incidental alteration of image data results in verification success as long as the alteration does not change image content from the human visual system point of view; in other words, the image perceptual meaning does not change. This verification is also often referred to as semi-fragile image (integrity) verification.

  Those fragile or semi-fragile image integrity verifications might identify locations in the image data/image content where the integrity is put into question. Solutions can include:

  1) Cryptographic methods such as Message Authentication Codes (MAC), digital signatures, cryptographic checksums or keyed hash.
  2) Watermarking-based methods. This Recommendation | International Standard does not define normative template for watermarking technology, although it supports non-normative tools using watermarking technology.
  3) Combination of the above two types of methods.

– Source authentication

  A JPSEC file can support a verification of the identity of a user/party which generated the JPSEC file. This can comprise methods of e.g., digital signatures or message authentication code (MAC).

– Conditional access

  A JPSEC file can support a mechanism and policy to grant or restrict access to image data or portions of those. This could allow for instance to view a low resolution (preview) of an image without being able to visualize a higher resolution.

– Registered Content identification

  A JPSEC file can be registered at a Content Registration Authority. It can support a method of matching the (claimed) image data/image content to the registered image data/image content. For example, such a method could be: Reading a file identifier (Licence Plate) which was placed inside the metadata, checking the coherence between this Licence Plate and the information that has been uploaded when the registration process was done. The Licence Plate might contain enough information to be able to request information from the Content Registration Authority where the file was registered and verify that the file corresponds to the identifier.

– Secure scalable streaming and secure transcoding

  A JPSEC file or sequence of packets can support methods such that the same or different nodes can perform streaming and transcoding without requiring decryption or unprotecting the content. An example is the case where protected JPEG 2000 content is streamed to a mid-network node or proxy that in turn transcodes the protected JPEG 2000 content in a manner that preserves end-to-end security.

5.3 Comments on design and implementation of secure JPSEC systems

This Recommendation | International Standard supports a rich and flexible set of security services. For example, the encryption primitives can be applied in a variety of different ways to achieve different goals, ranging from encryption of the entire JPEG 2000 codestream to selective encryption of only a small portion of the codestream. However, it is important to stress that significant care needs to be taken when implementing any security system, including one based on JPSEC.

It is strongly recommended that the designers of any security system carefully consider the recommended guidelines (Annex A) for the security primitives that are being employed. For most of the security primitives signalled using JPSEC, the associated ISO/IEC standards provide important guidance on their correct use. For example, for encryption using a block cipher and an associated block cipher mode (Table 29), guidelines for block cipher mode choice and operation are given in ISO/IEC 10116.

In addition, in many security applications authentication is the most important security service. Even when confidentiality is the targeted security service, it should be augmented by authentication to prevent various forms of attacks. Specifically, even in many imaging applications where the primary goal is confidentiality, it is recommended that authentication also be employed.

Key management is outside the scope of JPSEC. However, its criticality needs to be stressed. Of paramount importance in any cryptographic system is the management of the cryptographic keys that control the operations. If these keys are compromised, then the security of the whole system is compromised and in such a way that the compromise is not necessarily detected. It is therefore imperative that the keys are generated, distributed, stored and destroyed at a security level that is at least equal to that of the data that it is protecting. Furthermore, since the chances a key is compromised increase over time, it is also imperative that keys only be used for a fixed key lifetime. For more information on the use and management of cryptographic keys, see ISO/IEC 11770.

As with all security systems, the use of cryptographic operations are completely opaque to the user. That is, the user should not be able to discover any information about the cryptographic operations except for the output. For example, the user should not be able to access information about why a cryptographic operation failed to produce an output. Similarly, a user should not be able to find out any extra information even if he/she resorts to measuring "side channels" such as timing and/or power analysis. In short, the user should not be able to notice any difference in any of the applications outputs, regardless of what the application is currently doing, for if this is not the case the resulting leakage of information can potentially compromise the security of the system.

To summarize, it is strongly recommended that the designer of any security system, including one based on JPSEC, pay special attention to the details of the system design to ensure a secure system.

5.4 Byte aligned segment (BAS)

5.4.1 Byte aligned segment

In order to provide extensible signalling for classes and modes, this Recommendation | International Standard uses a variable length data structure called a "byte aligned segment" (BAS). Parameter fields with an extensible number of fields are represented with the Field BAS (FBAS) structure. Parameter values with large ranges are represented extensively with the Range BAS (RBAS) structure.

As illustrated in Figure 2, the BAS is composed of a sequence of one or more BAS bytes. The most significant bit (MSB) of each BAS byte indicates the existence of a following BAS byte. Specifically, if MSB = 1 then a subsequent BAS byte follows, while if MSB = 0 then a subsequent BAS byte does not exist, and the BAS structure is terminated. The remaining least significant bits of each BAS byte are concatenated to form a list of bits which are used in different ways for different BAS parameters. Often, they are used in conjunction with a parameter list that has a number of elements, and each BAS bit is set to 1 or 0 to flag information about its corresponding element. This flexible structure was chosen because of its extensibility for future evolutions of the standard, since it allows new parameters to be signalled in an extensible way.

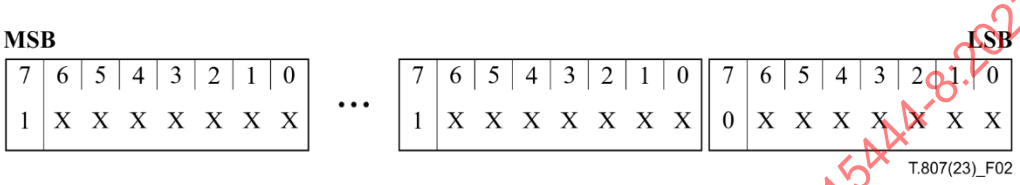


Figure 2 – Byte aligned segment (BAS) structure

5.4.2 Field BAS (FBAS)

A Field BAS (FBAS) is a type of BAS where the remaining bits of the BAS bytes are used to set fields to 1 or 0. An example of FBAS usage is the description class of the zone of influence (DCzoi), where multiple image descriptions such as tile index, resolution level, and colour component can be specified. If this is done, the three BAS bits corresponding to tile, resolution, and colour would be flagged to 1.

For example, to represent a Field BAS with 9 fields, f1 through f9, at most two BAS bytes would be needed. If the two bytes were byte "a" and byte "b", and the most significant bit of each byte were a0 and b0, then the FBAS would look like:

a0 a1 a2 a3 a4 a5 a6 a7 | b0 b1 b2 b3 b4 b5 b6 b7

a0 and b0 are the indicator bits. Field f1 through f7 are represented in bits a1 through a7, and field f8 is in bit b1 and field f9 is in bit b2. The remaining bits b3 through b7 are reserved and set to 0.

a0 f1 f2 f3 f4 f5 f6 f7 | b0 f8 f9 0 0 0 0 0

When used in a JPSEC stream, the FBAS in this example can be represented with one or two bytes, depending on the actual values of the field. This stems from the fact that the default value of the fields is 0. Thus, if fields f8 and f9 are not set (i.e., their value is 0), then the second byte of the BAS is not needed, and a0 is set to 0. On the other hand, if field 8 or field 9 is set, then two bytes are needed. In this case, a0 is set to 1 and b0 is set to 0.

Notice that the field bits are "left aligned". This allows us to add more fields over time in a compatible manner.

5.4.3 Range BAS (RBAS)

The Range BAS (RBAS) is used to extend the range or the number of bits used to represent a value. There are two types of RBAS, RBAS-8 and RBAS-16.

The RBAS-8 contains one or more RBAS bytes that contain the bits of the value. As in the FBAS, the first bit of each byte indicates whether another RBAS byte follows.

Unlike the FBAS, the RBAS is "right aligned". Thus, if a value has 9 significant bits v1 through v9, where v1 is the most significant bit, then it would be represented with two BAS bytes:

a0 a1 a2 a3 a4 a5 a6 a7 | b0 b1 b2 b3 b4 b5 b6 b7

as follows:

1 0 0 0 0 0 v1 v2 | 0 v3 v4 v5 v6 v7 v8 v9

If the value was small such that bits v1 and v2 were zero, then the two-byte representation above could be used with v1 and v2 set to zero, or a one-byte RBAS could be used as follows:

0 v3 v4 v5 v6 v7 v8 v9

The RBAS-16 can be used to represent values that are typically more than 7 bits but less than 15. In this case, the first RBAS chunk is two bytes where the first bit is the indicator and then next 15 bits are value bits, then the remaining bytes extended one byte at a time using the typical BAS structure where the first bit of each byte is the indicator of following BAS bytes.

a0 a1 a2 a3 a4 a5 a6 a7 | b0 b1 b2 b3 b4 b5 b6 b7 | c0 c1 c2 c3 c4 c5 c6 c7

If a parameter value had 22 bits, then it could be represented with the three-byte RBAS-16 structure shown below, where a0 and c0 are indicator bits to specify whether a BAS byte follows. Any remaining BAS bytes are traditional one-byte BAS segments.

a0 v1 v2 v3 v4 v5 v6 v7 | v8 v9 v10 v11 v12 v13 v14 v15 | c0 v16 v17 v18 v19 v20 v21 v22

Thus, the indicator bits would be set as follows:

1 v1 v2 v3 v4 v5 v6 v7 | v8 v9 v10 v11 v12 v13 v14 v15 | 0 v16 v17 v18 v19 v20 v21 v22

For both the RBAS-8 and RBAS-16, the value bits are also "right aligned".

Note that when writing JPSEC creators and consumers, it is important to pay attention to the big endian/little endian representations.

5.5 Main security marker (SEC)

5.5.1 Security marker segments

In this clause, a simple and flexible, yet powerful syntax for JPSEC signalling is presented. SEC marker segments are defined for this purpose and are located in the main header. The SEC marker segment syntax allows for the description of all required information for securing JPEG 2000 images. To do so, it makes references to JPSEC normative tools that are specified by the templates described in 5.8 or by JPSEC non-normative tools that can be defined *a priori* privately, and it makes provisions for handling parameters related to these tools.

A JPSEC codestream can be protected with one or more JPSEC tools. Each tool is a JPSEC normative tool or a JPSEC non-normative tool. The parameters for these tools are signalled in one or more SEC marker segments located in the main header of the codestream after the SIZ marker segment. When multiple SEC marker segments are used, they are concatenated and shall appear consecutively in the main header. In most cases, all the JPSEC parameters can be signalled in one SEC marker segment. However, in some cases the length of the signalling can exceed the maximum marker segment size. When this occurs, additional SEC marker segments can be used for signalling.

Figure 3 shows the syntax of the SEC marker segment. The segment is signalled by the SEC marker 0xFF65. L_{SEC} is the length of the SEC marker segment, including the 2 bytes for L_{SEC}, but not the two bytes for the SEC marker itself. Z_{SEC} is a SEC marker segment index. Z_{SEC} shall be set to 0 for the first marker segment that appears in the codestream. P_{SEC} is a parameter field that describes the security parameters relevant to the entire codestream and only exists in the first SEC marker segment, i.e., if $Z_{SEC} = 0$. The syntax supports the use of several JPSEC tools that are signalled in one or more marker segments. If more than one JPSEC tool is used, then a JPSEC consumer shall process the tools in the order in which they appear in the codestream.

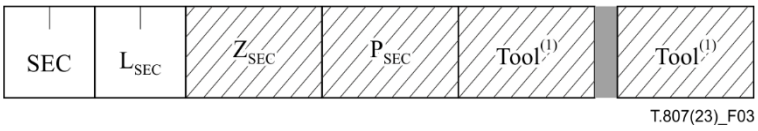


Figure 3 – Main security marker segment syntax

SEC: Marker code. Table 1 shows the sizes and values of the symbols and parameters for the main security marker segment.

L_{SEC}: Length of marker segment in bytes (including L_{SEC} itself, but excluding the marker).

Z_{SEC}: Index of this marker segment relative to all other SEC marker segments present in the current header. This field uses the RBAS structure.

P_{SEC}: Parameter field for codestream security parameters. This field is only present in the first SEC marker segment, i.e., when Z_{SEC} is 0.

$Tool^{(i)}$: Parameters for JPSEC tool *i*. If multiple JPSEC tools are signalled, then a JPSEC consumer shall process each tool in the order of appearance in the JPSEC codestream.

Table 1 – Main security parameter values

Parameter	Size (bits)	Values
SEC	16	0xFF65
L_{SEC}	16	2 ... (2^{16} – 1)
Z_{SEC}	8 + 8 * n (RBAS)	0 ... 2^{7+7*n}
P_{SEC}	0, if Z_{SEC} > 0 Variable, otherwise	If Z_{SEC} = 0, see Table 2
$Tool^{(i)}$	Variable	See 5.6.2 and 5.6.3

Figure 4 shows the syntax of the security parameters in the main header when multiple SEC marker segments are used. In this case, the JPSEC tool parameters are in different SEC marker segments. Each marker segment begins with the SEC marker, 0xFF65, and is followed by the length and index of the marker segment. The index of the first marker segment shall be set to 0 and shall increase by one for each marker segment in the order it appears. Only the first marker segment contains the security parameters for the codestream, P_{SEC}. All the marker segments contain the parameters for one or more JPSEC tools.

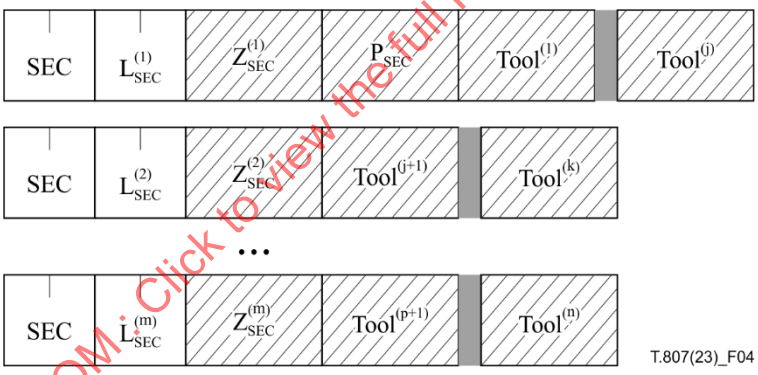


Figure 4 – Main security marker syntax when multiple marker segments are used

If needed, a JPSEC tool description can span multiple SEC marker segments, e.g., this can occur if the length exceeds the maximum SEC marker size. Since the length of the tool description is completely specified, the JPSEC creator simply splits the tool across SEC marker segments. The decoder should then concatenate all segments, minus the SEC marker and the L_{SEC} and Z_{SEC} values, and then interpret the tools accordingly.

P_{SEC} is a parameter field that describes security parameters for the entire codestream as opposed to for a particular tool. This is used to indicate events such as JPEG 2000 Core coding system compliance or the use of INSEC markers. The P_{SEC} parameters are shown in Figure 5.

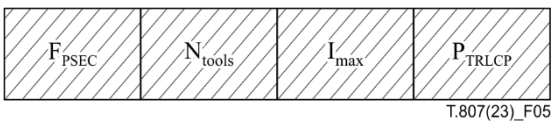


Figure 5 – Codestream security parameters (P_{SEC}) syntax

F_{PSEC}: Flag to indicate if INSEC marker segment is used, if multiple SEC marker segments are used, if the original JPEG 2000 Core coding system codestream data was modified, and if TRLCP tag usage is defined. FBAS structure is used by this field.

N_{tools}: Number of JPSEC tools used in the codestream. This field uses the RBAS structure.

I_{max}: Maximum tool instance index value used in the codestream. This field uses the RBAS structure.

P_{TRLCP}: Parameter field to define the format of TRLCP tag. This field exists if $F_{TRLCP} = 1$.

Table 2 – Codestream security parameters (P_{SEC}) in first SEC marker segment

Parameter	Size (bits)	Values
F_{PSEC}	Variable (FBAS)	See Table 3
N_{tools}	8 + n * 8 (RBAS)	1 ... 2^{7+7*n}
I_{max}	8 + n * 8 (RBAS)	0 ... 2^{7+7*n}
P_{TRLCP}	0, if $F_{TRLCP} = 0$ 32, if $F_{TRLCP} = 1$	See Table 4

F_{PSEC} is an FBAS structure used to indicate a number of parameter flags about the JPSEC codestream. The fields represented by F_{PSEC} are shown in Table 3. F_{INSEC} shall be set to 1 if INSEC markers are used in the JPSEC codestream. $F_{multiSEC}$ shall be set to 1 if multiple SEC marker segments are used in the JPSEC codestream. F_{mod} shall be set to 1 if the original JPEG 2000 data was modified in the JPSEC codestream. Note that if INSEC markers are used, then the original JPEG 2000 data is modified and thus F_{INSEC} and F_{mod} shall be set to 1. F_{TRLCP} shall be set to 1 if the TRLCP tag usage is defined in P_{SEC}. If it is defined, then the TRLCP tag descriptor, P_{TRLCP}, is specified in the P_{SEC} parameter field. The TRLCP tag usage shall be specified if any tool within the JPSEC codestream uses TRLCP tags.

Table 3 – Semantics for F_{PSEC} values (FBAS)

BAS field	BAS bit number	Value (bits)	Semantics
F_{INSEC}	1	0	INSEC is not used
		1	INSEC is used
$F_{multiSEC}$	2	0	One SEC marker segment is used
		1	Multiple SEC marker segments are used
F_{mod}	3	1	Original JPEG 2000 data was modified
		0	Otherwise
F_{TRLCP}	4	0	TRLCP tag usage is not defined in P_{SEC}
		1	TRLCP tag usage is defined in P_{SEC}

JPSEC defines a structure called a TRLCP tag that can be used to uniquely identify a JPEG 2000 packet. A JPEG 2000 packet can be uniquely specified by its tile index, resolution level index, layer index, component index, and precinct index. A TRLCP tag is defined as a data unit with a fixed number of bits used to specify each of these index values. The number of bits for each index is set in P_{SEC}. P_{TRLCP} is a parameter field that describes the format of the TRLCP tag as it shall be used in the JPSEC tools. This field only exists if $F_{TRLCP} = 1$. P_{TRLCP} consists of the following variables in Figure 6.

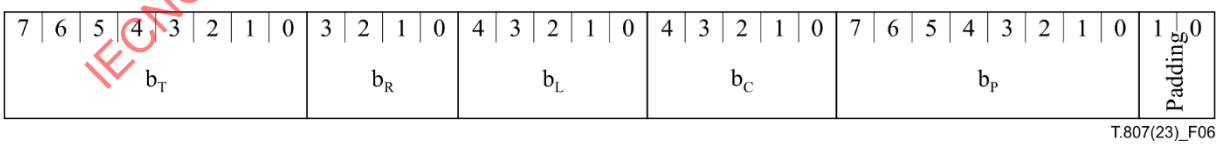


Figure 6 – TRLCP tag descriptor (P_{TRLCP}) syntax

b_T: Number of bits to represent tile index is $b_T + 1$ in TRLCP tag.

b_R: Number of bits to represent resolution level index is $b_R + 1$ in TRLCP tag.

b_L: Number of bits to represent layer index is $b_L + 1$ in TRLCP tag.

b_C: Number of bits to represent component index is $b_C + 1$ in TRLCP tag.

b_P: Number of bits to represent precinct index is $b_P + 1$ in TRLCP tag.

Table 4 – Parameter field for TRLCP tag descriptor (P_{TRLCP})

Parameter	Size (bits)	Values
b_T	8	0 ... (2^8 – 1)
b_R	4	0 ... 15
b_L	5	0 ... 31
b_C	5	0 ... 31
b_P	8	0 ... (2^8 – 1)
Padding	2	0

The size of each resulting TRLCP tag is the smallest integer byte size that contains all the bits. The format of the TRLCP tag contains the bits for the tile index, the resolution level index, the layer index, the component index, and the precinct index in that order. If extra bits are needed to fill the integer byte size requirement, then the TRLCP tag will be placed in the least significant bits possible, and the extra bits are set to 0. Note that these extra bits will be the MSBs of the TRLCP tag if they exist.

5.5.2 Application of multiple JPSEC tools

In many applications multiple JPSEC tools apply to a single JPEG 2000 codestream. For example, both encryption and authentication can be applied to protect a JPEG 2000 image. The general situation of applying multiple JPSEC tools is illustrated in Figures 3, 4 and 7, where N tools are applied. The JPSEC consumer will read the N tools in order of placement in the SEC marker segment shown in Figure 3 or Figure 4, and apply them in that same order to perform the JPSEC consumption of the JPSEC codestream. Note that while the JPSEC consumer applies the JPSEC tools in order 1, 2, ... , N, as read from the SEC marker segment, during creation of the JPSEC codestream these JPSEC tools were applied in the reverse order, i.e., N, N – 1, ...2, 1, as illustrated in Figure 7. Note that the numbering of the tools in the figure was chosen to highlight that the JPSEC consumer applies the JPSEC tools in the reverse order from the JPSEC creator. However, any numbering of the JPSEC tools is acceptable, as long as each JPSEC tool in a JPSEC codestream is given a unique number for identification purposes.

Generally speaking, JPSEC tools are created and consumed in reverse order of one another. For example, if the JPSEC creator applies N JPSEC tools, then the JPSEC consumer typically applies the same N JPSEC tools but in the reverse order. Correct JPSEC consumption of multiple JPSEC tools can be guaranteed by sequential consumption of the N tools in the correct order and by requiring any intermediate stage at the consumer to match the corresponding state at the creator. For example, in Figure 7, the state at the consumer after JPSEC consumption of tool 1 should be equal to the state after applying tool 2 during the JPSEC creation process. As a specific example of the state, the byte ranges should be consistent, therefore any bytes added when applying tool 1 should be removed when removing tool 1 at the JPSEC consumer.

In certain applications, it can be desirable for a JPSEC consumer to consume the multiple JPSEC tools in a different manner than described above. For example, the JPSEC consumer can choose to consume the multiple tools in a different order, or to skip certain tools in the consumption. Furthermore, the JPSEC consumer can prefer to apply certain JPSEC tools, but not remove them, e.g., to check a digital signature but not remove it. Careful consideration should be given in these cases to ensure that the out-of-order or skipped processing does not lead to incorrect or unintended consequences. This behaviour is not recommended unless the JPSEC application is fully aware of the potential ramifications.

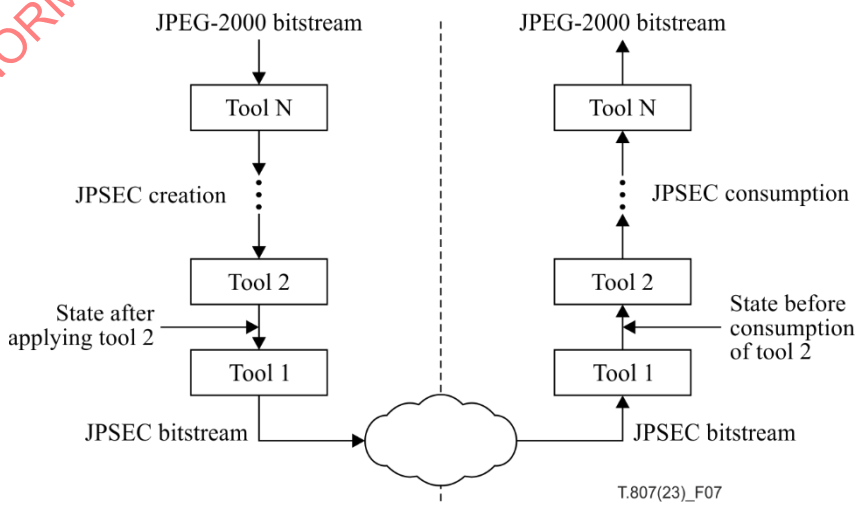


Figure 7 – Use of multiple JPSEC tools

5.6 JPSEC tools

5.6.1 JPSEC tool syntax

As mentioned earlier, there are two types of JPSEC tools. JPSEC normative tools are specified with the protection method templates described in 5.8, and are also known as JPSEC normative tools. JPSEC non-normative tools are specified by a particular JPSEC application based on their ID number, and are respectively referred to as JPSEC user-defined tools. The syntax for JPSEC normative tools are discussed in 5.6.2. The syntax for JPSEC non-normative tools are discussed in 5.6.3.

The syntax for JPSEC tools is shown in Figure 8. The JPSEC tool syntax has three main parts that describe:

1) what tool is applied with its identification;
2) where the tool is applied with a zone of influence structure; and
3) how the tool is applied with a more detailed parameter field.

For example, using this syntax, a JPSEC tool syntax could specify that a decryption tool should be used (what) on the lowest resolution component located in a particular byte range (where) using AES decryption in CBC mode with a specified set of initialization vectors and keys (how).

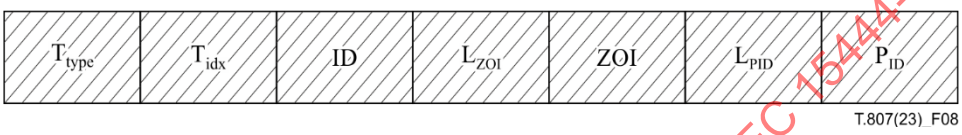


Figure 8 – JPSEC tool syntax (Tool$^{(i)}$)

T_{type}: Tool type. The value of 0 for the first BAS bit indicates a JPSEC normative tool. The value of 1 for the first BAS bit indicates a JPSEC non-normative tool. This field uses the FBAS structure.

T_{idx}: Tool instance index (can be used as a unique identifier). This field uses the RBAS structure.

ID: Identification value for JPSEC tool *i*. For JPSEC normative tools, the ID = ID_T is 8 bits and specifies the template type. For JPSEC non-normative tools, the ID = ID_{RA} is defined by Figure 10 and Table 8.

L_{ZOI}: Length of ZOI in Bytes (excluding L_{ZOI}). This field uses the RBAS structure.

ZOI: Zone of influence for JPSEC tool *i*.

L_{PID}: Length of P_{ID} in Bytes (excluding L_{PID}). This field uses the RBAS structure.

P_{ID}: Parameters for JPSEC tool *i*.

Table 5 – JPSEC tool parameter values

Parameter	Size (bits)	Values
T_{type}	8 + 8 * n (FBAS)	x0xx xxxx $_b$, x1xx xxxx $_b$
T_{idx}	8 + 8 * n (RBAS)	0 ... (2^{7+7*n} – 2) (2^{7+7*n} – 1), reserved
ID	8, if T_{type} = 0 Variable, if T_{type} = 1	See Table 6 See Figure 10 and Table 8
L_{ZOI}	16 + 8 * n (RBAS)	0 ... 2^{15+7*n}
ZOI	Variable	See 5.7
L_{PID}	16 + 8 * n (RBAS)	0 ... 2^{15+7*n}
P_{ID}	Variable	Table 7, if T_{type} = 0

Each JPSEC tool has the following syntax. The initial one-byte identifies if the tool is a JPSEC normative tool or JPSEC non-normative tool and assigns an instance identifier to the tool. This is followed by the tool identifier **ID**. This is followed by L_{ZOI}, which indicates the length of the subsequent zone of influence field ZOI, and the zone of influence itself, which describes where in the data stream the JPSEC tool is applied. This is followed by L_{PID}, which indicates the length of the following parameter field P_{ID}, which is a field to transmit one or more parameters for the JPSEC tool.

The first byte of the tool uses a one-byte FBAS structure whose first BAS bit represents the tool type, T_{type}, where 0 specifies a JPSEC normative tool and 1 specifies a JPSEC non-normative tool. This is followed by the instance index, T_{idx}, which is represented using the RBAS structure. The instance index shall be a unique identifier of the tool within the

codestream, and thus shall not be repeated by any other tool in the codestream, even if it is in a different SEC marker segment. The instance index is especially critical (and necessary) when INSEC markers are used, because each INSEC marker segment contains the instance index of the tool to which it applies. It is recommended that the first tool applied at a JPSEC creator has an instance index of 1, and that each additional tool be indexed sequentially as it is applied at the protector.

In addition, each JPSEC tool has an ID number which is 8 bits for JPSEC normative tools and 32 bits for JPSEC non-normative tools. For JPSEC normative tools, the ID number describes which protection method template is used, i.e., it specifies the decryption template, authentication template, or hash template. For JPSEC non-normative tools, the first bit indicates whether it is a JPSEC user-defined tool. In either case, the ID number indicates the particular tool. However, a JPSEC application that uses user-defined ID numbers runs the risk of choosing an ID number that is also used by another JPSEC application, so this should be used cautiously.

When each JPSEC tool is applied at the JPSEC creator, the P_{SEC} parameter field shown in Table 2 shall be updated. For example, the P_{SEC} parameter field contains the I_{max} parameter that specifies the maximum instance index used for the tools in the JPSEC codestream. When a new tool is applied, it shall be assigned a unique instance index. A JPSEC protector can refer to the I_{max} parameter given in the P_{SEC} parameter field to determine the instance index to assign to a JPSEC tool, for example, it can choose a value that is one greater than the current I_{max} value, and it should then increment the value of I_{max} by 1 accordingly.

5.6.2 JPSEC normative tool

The JPSEC normative tool uses the JPSEC tool syntax described in 5.6.1 and shown in Figure 8, where the tool type $T_{type} = 0$ and the size of the ID is 8 bits. JPSEC normative tools are based on the protection method templates described in 5.8. There are three types of protection method templates; the type used by the tool is specified by the tool identifier ID = ID_T using the values shown in Table 6.

Table 6 – JPSEC normative tool Template ID values (ID_T)

Values	Protection method template
0	Reserved
1	Decryption template
2	Authentication template
3	Hash template
4	NULL tool
	All other values are reserved for ITU-T | ISO use

In the case of JPSEC normative tools, the parameter field P_{ID} has the structure shown in Figure 9. P_{ID} consists of four main fields: the protection method template T, its processing domain PD, its granularity G, and its value list V. The syntax for each of these fields is given in 5.8, 5.9, 5.10 and 5.11, respectively. Together, these fields describe how the tool is applied. The protection method template T describes the particular protection method for the decryption template, authentication template, or hash template specified by the normative tool ID. It can also specify the NULL tool, in which case no template is used, but other functionalities can still be used. For example, the zone of influence can be specified to represent image regions and their corresponding byte ranges. The processing domain PD describes the domain in which the protection method is applied. The granularity G describes the granularity with which the protection method is applied. The value list V contains a list of values that can be needed by each protection method with finer granularity. For the decryption template, the value list can be used to specify a finer grain set of initialization values that are to be used. For the authentication template, the value list contains a set of MAC values or digital signatures. For the hash template, the value list contains a set of hash values. In all cases, the value list contains a granularity of values specified by the granularity field G.

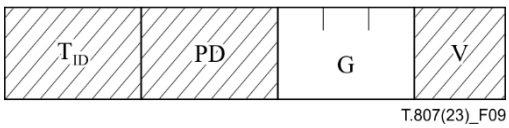


Figure 9 – Parameters (P_{ID}) syntax for JPSEC normative tools (t = 0)

T_{ID}: Template parameters for JPSEC normative tool with template identifier ID_T.

PD: Processing domain for JPSEC normative tool.

G: Granularity for JPSEC normative tool.

V: Value list for JPSEC normative tool, e.g., initialization vectors, MAC values, digital signatures, or hash values depending on template ID.

Note that the template parameters depend on the template ID. However, the processing domain, granularity, and value list are independent of the template ID.

Table 7 – JPSEC normative tool parameter values

Parameter	Size (bits)	Values
T_{ID}	0, if $ID_T = 4$ Variable, otherwise	N/A See 5.8
PD	Variable	See 5.9
G	24	See 5.10
V	Variable	See 5.11

5.6.3 JPSEC non-normative tool

The mechanism to register JPSEC user defined tool to JPSEC RA tool described in this clause is deprecated because the registration authority defined in the previous edition, ISO/IEC 15444-8:2007, is cancelled.

In certain cases, it can be useful for a JPSEC application to have the ability to apply a tool that extends beyond the JPSEC normative tools. This capability is supported by using a JPSEC user defined tool which was called as JPSEC non-normative tool. This enables one to use many elements of JPSEC normative tools, including the ZOI and the JPSEC templates, but adds the flexibility of using the parameters in a different manner associated with a tool ID value.

The JPSEC non-normative tool used the JPSEC tool syntax described in 5.6.1 and shown in Figure 8, where the tool type $T_{type} = 1$ and the identifier ID_{RA} consists of a name space and an ID number, as defined by Figure 10 and Table 8.

There were two classes of JPSEC non-normative tools:

1) JPSEC registration authority tools (Deprecated): JPSEC non-normative tools whose signalling was specified with a registration authority.
2) JPSEC user-defined tools: JPSEC non-normative tools whose signalling is specified by a JPSECapplication.

These two classes of JPSEC non-normative tools were signalled using the 32-bit $ID_{RA,id}$ identifier shown in Table 9, where the identifiers whose first bit is a 0 were defined by a registration authority, and those whose first bit is a 1 are defined by a particular JPSEC application.

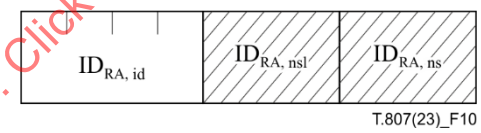


Figure 10 – ID_{RA} syntax

$ID_{RA,id}$: Tool identifier for user-defined tool

$ID_{RA,nsl}$: Length of the field $ID_{RA,ns}$ in bytes. This field uses RBAS.

$ID_{RA,ns}$: A string containing the name space of the specified user-defined tool

Table 8 – Parameters values in ID_{RA} syntax

Parameter	Size (bits)	Values
$ID_{RA,id}$	32	See Table 9
$ID_{RA,nsl}$	8 + 8 * n (RBAS)	0 ... $(2^{7+7*n} - 1)$
$ID_{RA,ns}$	Variable	A string containing namespace

Table 9 – ID values for JPSEC non-normative tools ($ID_{RA,id}$)

$ID_{RA,id}$	Meaning
0x00 00 00 00 ... 0x7F FF FF FF	Previously managed by JPSEC RA (Registration Authority) tool. Values were to be managed by JPSEC registration authority (Deprecated). Reserved for ITU-T | ISO use.
0x80 00 00 00 ... 0xEF FF FF FF	JPSEC user-defined tool. Values can be defined by a particular JPSEC application.
0xF0 00 00 00 ... 0xFF FF FF FF	Reserved for ITU-T | ISO use.

For RA tools, the field $ID_{RA,ns}$ was the name space of the Registration Authority (RA) with which this tool is registered. As each RA had a unique name space, the $ID_{RA,id}$ and $ID_{RA,ns}$ were used together to identify an RA tool. For user-defined tools, the field $ID_{RA,ns}$ is chosen by the developers. In order to limit the risk of ID collisions, it is recommended that the developers seek uniqueness when choosing their name space, for example, by choosing the domain name of their organization or company. However, note that for user-defined tools, there is no way to guarantee that uniqueness of the name space, so ID collision can occur and should be carefully considered when using user-defined tools.

The P_{ID} field is used to transmit one or more parameters for the JPSEC non-normative tool *i*. The format of the P_{ID} field is not fully given in the scope of JPSEC. If a user-defined is used, then only the length of this field is specified, and it is up to the users to appropriately use this field.

However, JPSEC does allow the syntactical structures defined for JPSEC normative tools to be used in the P_{ID} field for JPSEC non-normative tools. For example, a JPSEC non-normative tool can use the protection method templates, processing domain, granularity, and value list fields described in 5.8, 5.9, 5.10 and 5.11, respectively.

This syntax is very flexible and can accommodate a wide variety of security techniques, such as image data integrity, access control and rights protection methods. Hence, it offers a rich set of functionalities while being simple and concise.

5.7 Zone of Influence (ZOI) syntax

5.7.1 Introduction

The Zone of Influence (ZOI) can be used to describe the coverage area of a JPSEC tool. The data within the coverage area (specified by the ZOI) is referred to as the influenced data. JPSEC normative tools shall use the ZOI to describe their coverage area. JPSEC non-normative tools can use the ZOI to describe their coverage area or they can use an alternative method. If an alternative method is used, then the ZOI length is 0, i.e., it does not exist.

The Zone of Influence (ZOI) describes the coverage area of each JPSEC tool. This coverage area can be described by image-related parameters, e.g., by resolution or image area; or by non-image related parameters, e.g., by codestream segments or packet indices. In cases where image related parameters and non-image related parameters are used together, the ZOI describes the correspondence between these areas. For example, the ZOI can be used to indicate that the resolutions and image area specified by the image related parameters correspond to the codestream segments specified by the non-image related parameters. This allows the ZOI to be used as metadata that signals where certain parts of the image are located in the JPSEC codestream.

Figure 11 illustrates the conceptual structure of the ZOI. The ZOI contains one or more zones. When multiple zones are used within a single ZOI, the ZOI is defined by their union. This indicates that the JPSEC tool should be applied to all the zones. Each zone in a ZOI is described by three fundamental units: description class, parameter mode and parameter items (values). This Recommendation | International Standard defines two description classes: image related description class and non-image related description class. These parameters can be specified using a number of modes, for example, by a single value, multiple listed values, or by a range. The parameter values or items are then listed in accordance with the mode.

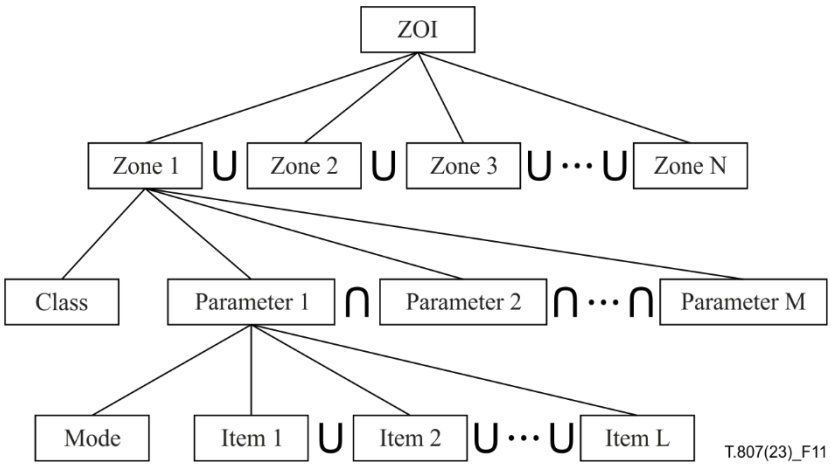


Figure 11 – Zone of Influence conceptual structure

5.7.2 ZOI syntax

Figure 12 shows the ZOI syntax. The ZOI can contain one or more zones. It can also be empty, in which case NZzoi shall be 0. When this occurs, the influence of the tool is specified by other means, such as by the INSEC marker or by parameters defined by a JPSEC non-normative protection tool.

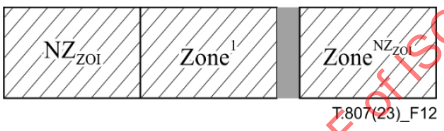


Figure 12 – ZOI syntax

NZzoi: Number of Zones. This field uses the RBAS structure.

Zone^k: Zone. Its structure is specified in 5.7.3.

Table 10 – Zone of influence field (ZOI) parameter values

Parameter	Size (bits)	Values
NZzoi	8 + 8 * n (RBAS)	0 ... (2^{7+7*n} – 2) (2^{7+7*n} – 2), reserved
Zone^k	Variable	See 5.7.3

5.7.3 Zone syntax

The Zone contains a zone description class field indicator followed by parameters of that class. The zone description class uses the FBAS structure. As shown in Figure 13, the second most significant bit in each byte, labelled "x", flags the use of a specific description class. This Recommendation | International Standard defines two description classes: image related description class and a non-image related description class (see Table 12). Tables 13 and 14 define the field indicator numbers for the image related description class and non-image related description class, respectively. The concatenation of the six bits labelled "y", in each byte that follow the description class flag, indicates the use of a specific description within a given description class. A bit value of "1" at a bit number in each class indicates that the corresponding parameter field exists. The number of parameters shall be the same as the number of zone description class field indicators set to '1', and shall appear in order which the class field indicator is signalled. The zone description class has variable number of bytes; when the MSB equals 1, then another zone description class byte follows. The MSB of the last description class byte equals 0. If both the image related and non-image related description classes are used, then the image related description class bytes shall precede the non-image related description class byte. When a number of items are represented using this structure, the first item in the list shall correspond to the most significant available bit of the first byte.

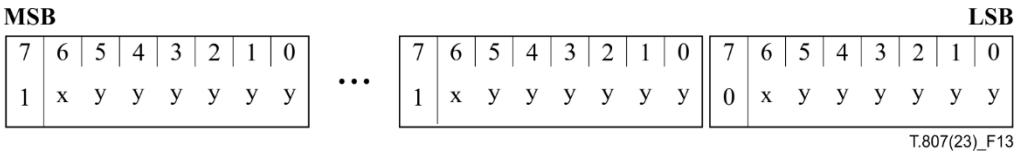


Figure 13 – Zone description class structure (DCzoi)

Figure 14 shows the Zone syntax.

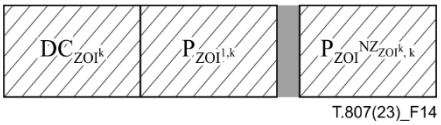


Figure 14 – Zone syntax consists of a description class and one or more parameter sets

DCzoi^{k}: kth Zone description class. This field uses the FBAS structure.

Pzoi^{i,k}: The Zone parameters according to the specified Zone description class (DCzoi^{k}). See 5.7.6.

DCzoi^{k} specifies the number *n* of zone description class fields that exist, based on the number of bits that are set to one. For each zone description class field, there exists one Pzoi^{i,k} zone parameter field. These fields appear sequentially in the same order that the flags appear in DCzoi^{k}.ZA

Table 11 – Zone parameter values

Parameter	Size (bits)	Values
DCzoi^{k}	Variable (FBAS)	Varies according to the value set in Table 12
Pzoi^{i,k}	Variable	See 5.7.6 for the syntax of this field

Table 12 – Description class indicator value

Value	Description class
0	Image related description class. The following bit numbers are defined in Table 13
1	Non-image related description class. The following bit numbers are defined in Table 14

Table 13 – Image related description class

Bit number	Semantics
1	Image region
2	Tile(s) as defined in the JPEG 2000 Core coding system
3	Resolution level(s) as defined in the JPEG 2000 Core coding system
4	Layer(s) as defined in the JPEG 2000 Core coding system
5	Component(s) as defined in the JPEG 2000 Core coding system
6	Precinct(s) as defined in the JPEG 2000 Core coding system
7	TRLCP (Tile-Resolution-Layer-Component-Precinct) tag(s)
8	Packet(s) as defined in the JPEG 2000 Core coding system
9	Sub-band(s) as defined in the JPEG 2000 Core coding system
10	Code-block(s) as defined in the JPEG 2000 Core coding system
11	ROI(s)
12	Bit-rate
13	User-defined. The details shall be specified by other means. (e.g., JPSEC ID)
	All other values are reserved

Table 14 – Non-image related description class

Bit number	Semantics
1	Packet(s) as defined in the JPEG 2000 Core coding system
2	(Padded) Byte range(s) (beginning at first byte after the first SOD marker)
3	(Padded) Byte range(s) (beginning at first byte after the first SEC marker)
4	Unpadded byte range(s) when padding is used
5	TRLCP (Tile-Resolution-Layer-Component-Precinct) tag(s)
6	Distortion value(s)
7	Relative importance(s)
8	User-defined. The details shall be specified by other means. (e.g., JPSEC ID)
	All other values are reserved

Packet indices are numbered sequentially within a tile, and therefore they are not necessarily unique across tiles. Furthermore, packet indices within a tile can roll over when their maximum value of 65535 is exceeded. For this reason, packet indexing is described in more detail. When the packet indices within a tile do not exceed 65535 packets, then the packet index described in Table 13 is defined by the packet index given by the SOP Nsop parameter as defined in Table A.40 in the JPEG 2000 Core coding system standard. Note that when the maximum value does not exceed 65536, a single JPEG 2000 packet can be specified uniquely with a tile index and a packet index. When the packet indices exceed 65535 packets, then the JPEG 2000 Core coding system packet index is defined to roll over to 0. In this case, the packet index does not uniquely identify a packet and shall not be used. In this case, it is recommended to use the TRLCP tag instead. Please note that security services which require unique packet indices are vulnerable if the packet index rolls over and repeats.

When the TRLCP tag is used, its format shall be defined in the P_{SEC} parameter field shown in Table 2. Specifically, the TRLCP tag format is specified by the P_{TRLCP} parameter field in Table 4. This defines the size of TRLCP tags in the ZOI.

The non-image related description class can also have multiple fields set simultaneously. When this occurs, the modes for the various parameter fields shall have the same number of items (one exception to this rule is described below), and these items shall correspond with one another in a one-to-one manner in the same order. For example, if the zone uses byte ranges and packet ranges, each should have the same number of range items where the first byte range corresponds to the first packet range, and so on.

There is one exception to the above rule on requiring the same number of items for each field. This occurs when one of the fields f1 contains 1 item which specifies a range of items (as described by the range mode in 5.7.6) where this range contains N elements and when another field f2 is specified by a list of N items. In this case, the field f1, which contains only 1 item (the range) is interpreted as a list of N items. These N items specified by the range in f1 shall correspond one-to-one with the N items listed in f2. Therefore, a range of items can be associated to either a single item or to multiple items (one for each item in the range).

The bytes are indexed either from the first byte after the first SOD marker or from the first byte after the first SEC marker. In either case, this first byte is labelled as byte 0.

The distortion fields (both the distortion and relative importance fields) provide the capability to signal the importance of areas specified by the ZOI. The distortion parameter specifies the distortion-reducing contribution of the specified data segment, be it for a set of packets or a byte range or for the specified image-related area. The distortion is expressed in terms of the total squared error, using either a one-byte or a two-byte description signalled in the Mzoi. The relative distortion parameter can be used to specify the relative importance of specified data segments, using either one-byte, two-byte, or four-byte values signalled in the Mzoi. Additional details and the formats of these fields is described in 5.7.3.2.

The TRLCP tag specifies a protected packet's tile, resolution, layer, component, and precinct in the codestream. This tag is used in the ZOI to specify these parameters because this information can be difficult to infer in a protected codestream.

Note that when only image-related descriptions are used, the field can be terminated. Thus, one does not need to represent non-image related descriptions if they are not used.

5.7.3.1 Byte range fields

The non-image-related description class allows the ZOI to be described in byte ranges. In general, the 2nd and 3rd elements of Table 14 should be used to represent the byte ranges for most tools such as authentication and encryption/decryption without padding. However, some protection methods, such as encryption/decryption with padding, change the length of the data. When this occurs, both the padded byte range and the unpadded or original byte range are specified. In this case, the padded byte range is specified by the 2nd or 3rd element of Table 14 according to the needs of the protection tool. (Note that these two elements cannot be used together.) In addition, the unpadded byte range is

specified by the 4th element of Table 14. The unpadded byte range should be specified with the same description mode as the padded byte range and have same number of items. These items should correspond to each other in a one-to-one manner in the same order.

5.7.3.2 Distortion field and relative importance field

The distortion and relative importance fields provide the capability to signal the importance of areas specified by the ZOI.

The distortion field is used to associate a distortion with an area specified by the ZOI. The distortion value specifies the total squared error (or sum of squared error) distortion that would result if the associated area is not available for decoding. Total squared error distortion is a basic distortion metric used in image and video processing, and it is used to derive the common mean-squared-error (MSE) distortion and peak-signal-to-noise (PSNR) ratio. The distortion field is expressed using a one-byte or a two-byte description, where these one-byte and two-byte descriptions are described below, and the choice of one-byte or two-byte description is signalled by the Mzoi parameter value which specifies the length of this field. The relative importance field can be used to describe the relative importance among different areas specified by associated ZOIs, without necessarily being tied to a specific distortion metric. The length of the relative importance field is also signalled by the Mzoi. These fields are discussed in more detail in the following.

5.7.3.2.1 One-byte distortion field

The total squared error distortion is expressed using a one-byte distortion field with a pseudo floating-point type representation. The 8 bits available in the distortion field are allocated as shown in Figure 15 and Table 15 to provide an appropriate trade-off between accuracy and dynamic range. Note that a sign bit is unnecessary since distortion is non-negative. To cover a sufficient dynamic range, base 16 is used and 4 bits are used for the exponent (exp). The mantissa (m) is expressed using 4 bits. Therefore, the total distortion value D is given by:

$$D = m \times 16^{exp}$$

where m has a value in the range $0 \le m \le 15$ and exp has a value in the range $0 \le exp \le 15$. A distortion value of zero is represented by m = 0 and exp = 0, that is by the distortion field being zero. By allocating 4 bits for the mantissa m the accuracy is within $\frac{1}{2} \times (1/2^4) = 1/32$ or about 3%. With 4 bits for the exponent and using base-16 the dynamic range is from 0 to max, where max is given by m = 15 and exp = 15 which corresponds to a distortion of $15 \times 16^{15} = 1.7 \times 10^{19}$.

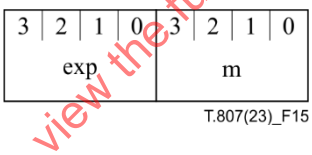


Figure 15 – Distortion field syntax

exp: Exponent of distortion field value (base 16)

m: Matissa of distortion field value

Table 15 – Distortion field parameter values

Parameter	Size (bits)	Values
exp	4	0 ... 15
m	4	0 ... 15

Note that with this format for the distortion, a comparison between two distortions to determine which is larger can be simply achieved by comparing the two distortion values as unsigned char. Specifically, to perform this comparison there is no need to convert from the pseudo floating-point format to the actual total distortion in order to determine which of two distortion values is larger or smaller. This property can simplify the processing in various applications.

5.7.3.2.2 Two-byte distortion field

In the two-byte format as shown in Figure 16, distortion values shall be expressed as a two-byte number in pseudo floating-point format. The pseudo floating-point format for distortion is defined as follows. This format is used in D.1.1.1 (Equation D.3) of Rec. ITU-T T.800 | ISO/IEC 15444-1 to express the quantization step size for JPEG 2000. Each 16-bit number contains the exponent (5 bits) and mantissa (11 bits) of the metric value. In particular, the floating-point value *V* of the metric is given by the following formula:

$$V = 2^{\varepsilon-15}\left(1+\frac{\mu}{2^{11}}\right) \quad \text{if } \varepsilon \neq 0$$

$$V = 0 \quad \text{if } \varepsilon = 0$$

where ε is the unsigned integer obtained from the first five most significant bits of the parameter, and μ the unsigned integer obtained from the remaining 11 bits. The special case of $V = \infty$ correspond to μ = 0 and ε = 31. Note that values that would underflow the representation are set to zero.

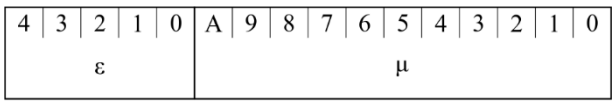


Figure 16 – Distortion field syntax

ε: Exponent of two-byte distortion field value.

μ: Matissa of two-byte distortion field value.

Table 16 – Distortion field parameter values

Parameter	Size (bits)	Values
ε	5	0 ... 31
μ	11	0 ... $(2^{11} - 1)$

The algorithm to compute ε and μ is not defined as a mandatory part of this Recommendation | International standard. A possible technique performs the following steps (an example of conversion of the number 12.25 is provided). If $V = 0$, set ε = μ = 0. Otherwise:

– convert V to a binary number ($12.25_{10} = 1100.01_2$);

– normalize the number; this means there should be a 1 digit to the left of the binary point and multiplication by the appropriate power of two to represent the original value. The normalized form of 1100.01 is 1.10001×2^3;

– the exponent is the power of 2, presented in excess notation. The exponent bias is 15; hence for this example the exponent is represented as 18_{10} (10010_2);

– the mantissa represents the significant bits, *except for the bit to the left of the binary point*, which is always one and therefore does not need to be stored; zeros are possibly appended so as to obtain 11 bits. For this example, the mantissa is 10001000000.

5.7.3.2.3 Relative importance field

The relative importance field *r* can be used to describe the relative importance among different coding units, without necessarily being tied to a specific distortion metric. This enables one to describe the relative importance or prioritization among coding units without explicitly describing how much more important one is from another. This relative importance of the associated data is specified by an n-byte field which supports 2^{8n} possible rankings as shown in Figure 17 and Table 16, where the number of bytes *n* for this field is specified by Mzoi. For example, by using a one-byte relative importance field a total of 256 possible rankings are supported. Increasing values correspond to increasing importance, in a similar manner to the distortion field.

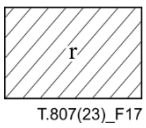


Figure 17 – Relative importance field syntax

r: Relative importance value

Table 17 – Relative importance field parameter values

Parameter	Size (bits)	Values
r	8 * n	0 ... $(2^{8n} - 1)$

5.7.3.2.4 Additional comments on distortion field and relative importance field

Since for both the one-byte distortion field and one-byte relative importance field the larger values correspond to greater importance, it is possible to make comparisons for these two data units in the same manner irrespective of whether the distortion field specifies actual distortion or a relative importance. This can simplify applications.

Headers can be specified using the distortion or relative importance fields. The loss of various types of data, such as the main and tile-part headers or the SEC header, prevent the decoding of the related image data. The JPSEC creator can assign distortion to this data using either:

1) the highest distortion value (specified next) to signal the header or critical data; or
2) to describe the total distortion that would be created if the image or portion of the image is undecodable.

The creator then has some flexibility in how to signal the headers.

The highest distortion value for the one-byte fields is a byte of all ones (0xFF). Note that this value is the highest possible distortion value for both the one-byte total squared error distortion field and for the one-byte relative importance field. The highest distortion value for the two-byte distortion field is the two bytes of all ones (0xFFFF). The highest importance for the relative importance field of length n-bytes is an n-byte value of all ones.

5.7.3.2.5 Joint use of distortion field and relative importance field

The distortion field and relative importance field can be used simultaneously to describe the area specified by the ZOI. In this case both fields specify squared-error distortion, however the distortion field specifies the incremental reduction in distortion while the relative importance field specifies the total distortion. Specifically, the distortion field specifies the incremental reduction in distortion that the ZOI would produce if decoded. This assumes that all information required to decode the ZOI is available, and focuses on the incremental reduction in distortion produced by the ZOI. The relative importance field specifies the total distortion that would be incurred if the ZOI is not available, i.e., it specifies the total distortion that would result if the given ZOI is unavailable for decoding by accounting not only for the value of the ZOI itself (as expressed by the distortion field) but also accounting for the distortion produced because other parts of the compressed bitstream which depend on the ZOI are undecodable. The total distortion associated with different ZOIs provides a useful metric for the relative importance of the different ZOIs. When both fields are used they will use the same mathematical expression for distortion, as signalled by the distortion field.

5.7.3.3 Bit-rate field

The Bit-rate field is used to specify the protected zone in wavelet coefficient domain. It identifies the most significant bit-planes whose compressed bit-rate is specified by this field. The MSBs are selected using the rate-distortion optimization process specified in the JPEG 2000 Core coding system. For example, if the Bit-rate value is 2.5, the protected zone includes the MSBs of all wavelet coefficients whose compressed bit-rate is 2.5 bit per pixel. The syntax of Bit-rate field is shown in Figure 18 and Table 18. The specified bit-rate is given by:

$$R = I_R + F_R/16$$

For example, a bit-rate of zero is represented by $I_R = 0$ and $F_R = 0$; a bit-rate value of 2.5 is represented by $I_R = 2$ and $F_R = 8$.

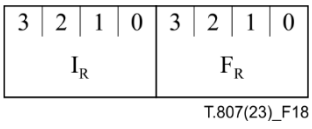


Figure 18 – Bit-rate field syntax

I_R: The integer part of the specified bit-rate.

F_R: The fractional part of the specified bit-rate.

Table 18 – Bit-rate field parameter values

Parameter	Size (bits)	Values
I_R	4	0 ... 15
F_R	4	0 ... 15

5.7.4 Relationship between multiple parameters

5.7.4.1 Global

When the image-related description class has multiple fields set simultaneously, the resulting zone shall be the intersection of these fields. For example, a zone could specify the lowest resolution level in the 2nd tile. The union of fields can be specified by using multiple zones in the ZOI.

The non-image related description class can also have multiple fields set simultaneously. When this occurs, the modes for the various parameter fields shall have the same number of items (one exception to this rule is described below), and these items shall correspond with one another in a one-to-one manner. For example, if the zone uses byte ranges and packet ranges, each should have the same number of range items where the first byte range corresponds to the first packet range, and so on.

There is one exception to the above rule on requiring the same number of items for each field. This occurs when one of the fields f1 contains 1 item which specifies a range of items (as described by the range mode in 5.7.6) where this range contains N elements and when another field f2 is specified by a list of N items. In this case, the field f1, which contains only 1 item (the range) is interpreted as a list of N items. These N items specified by the range in f1 shall correspond one-to-one with the N items listed in f2. Therefore, a range of items can be associated to either a single item or to multiple items (one for each item in the range).

5.7.4.2 Examples

As previously illustrated in Figure 11, the zone description class structure can have multiple fields set simultaneously, where N fields are image related descriptions (D_i^1, D_i^2, ... , D_i^N) and M fields are non-image related descriptions (D_n^1, D_n^2, ... , D_n^M). The semantics can be understood as $\{D_i^1 \cap D_i^2 \cap \ldots \cap D_i^N\} = D_n^1 = D_n^2 = \ldots = D_n^M$, that is, the intersection of the N image related descriptions is corresponding to each of the M non-image related descriptions, and in addition, the M non-image related descriptions are mutually corresponding to each other. This relationship is further illustrated with three examples below.

In the first example, the zone description has two image related descriptions: one for resolution 2 and the other for layer 3. In this case, the influenced data is the intersection of resolution 2 and layer 3, as illustrated in Figure 19.

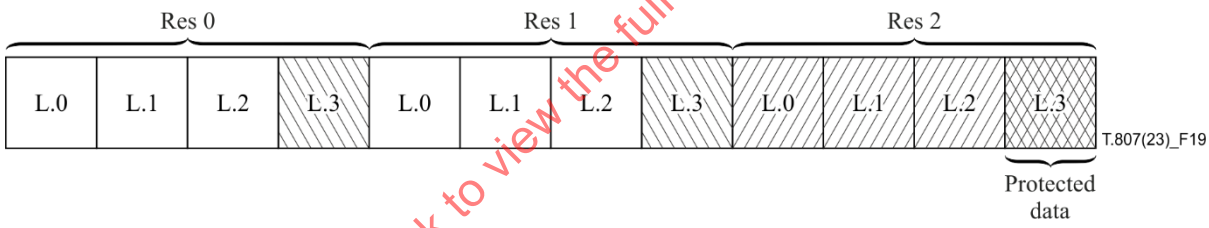


Figure 19 – ZOI example using image related descriptions

In the second example as illustrated in Figure 20, the zone description has two image related descriptions (which are resolution 2 and layer 3) and one non-image related description (which is packet range 80-100). In this case, the influenced data is the intersection of resolution 2 and layer 3. Furthermore, this indicates that the influenced data is contained in packets ranging from 80 to 100.

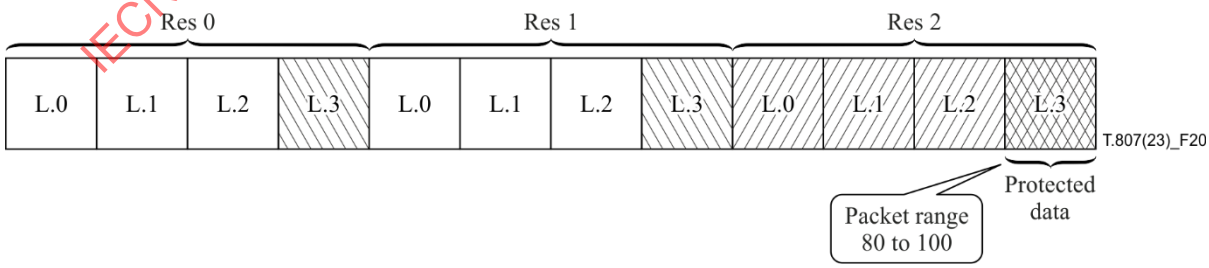


Figure 20 – ZOI example using image related and non-image related descriptions

In the third example as illustrated in Figure 21, the zone description has two image related descriptions (which are resolution 2 and layer 3) and two non-image related descriptions (which are packet range 80-100 and byte range 856-1250). Once again, the influenced data is the intersection of resolution 2 and layer 3, and the influenced data is

contained in packets ranging from 80 to 100. Furthermore, these packets and influenced area are located in the byte range 856-1250.

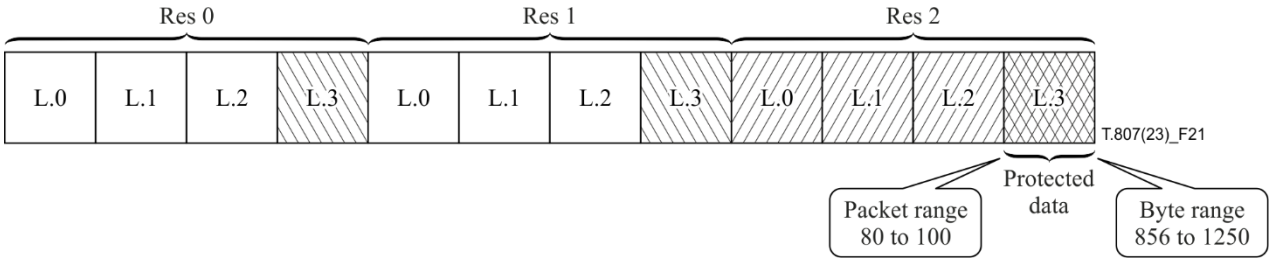


Figure 21 – A second ZOI example using image related and non-image related descriptions

5.7.5 Protecting any data that follows the SEC marker

The above discussion has largely focused on supporting protection services for the JPEG 2000 codestream. However, many elements of the main header, including JPSEC signalling, should also be protected, and the ZOI and protection methods can also be used for this purpose.

Specifically, the byte range mode of the non-image related description class can be used to specify that a JPSEC tool should be applied to any data following the SEC marker. As described before, the first byte of the SEC header is byte 1 for indexing the byte range. The data that follows the SEC marker and that can be protected includes the SEC segment and most of the main header. Note that all of the JPEG 2000 main header, except for the SIZ marker segment, can be moved after the SEC marker and hence can be protected using the above approach. If the JPEG 2000 SIZ marker segment is to be protected, it can be done at a higher level, e.g., file format layer.

The JPSEC tools for protecting the SEC segment should generally be the first tools in the SEC segment. This enables the consumer to first render the SEC segment data, which can then be used to process the remainder of the codestream.

5.7.6 Zone description parameter syntax (Pzoi)

Figure 22 shows the ZOI description parameter syntax.

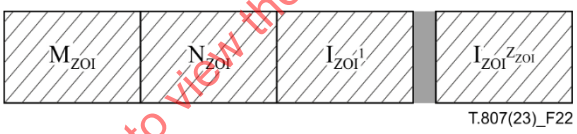


Figure 22 – ZOI description parameter syntax

Mzoi: ZOI description mode. This field uses the FBAS structure.

Nzoi: Number of Izoi. This field uses RBAS structure.

Izoi^{i}: Item.

Table 19 – Pzoi^{i} parameter values

Parameter	Size (bits)	Values
Mzoi	Variable (FBAS)	See Table 20
Nzoi	0 8 + 8 * n (RBAS)	If bit number 2 of Mzoi is 0. 2 ... (2^{7+7*n} – 1)
Izoi^{i}	Variable	Depends on the mode specified in Mzoi

Table 20 – Mzoi parameter values

FBAS bit number	Values (bits)	Semantics
1	0	The specified zones are influenced by the JPSEC tool
	1	The complement of specified zones are influenced
2	0	Single item is specified
	1	Multiple items are specified
3, 4	00	Rectangle mode. A rectangle region where the first value pair specifies the upper-left corner and the second value pair specifies the lower-right corner such that both corners are inclusive. For each corner, the first value shall be the horizontal position and the second value shall be the vertical position. The indexing shall begin at 0, and shall use the reference grid defined in the JPEG 2000 Core coding system.
	01	Range mode. A range of values where the first value specifies the start index and the second value specifies the last index, both inclusive.
	10	Index mode. Specifies single value(s).
	11	Max mode. Specifies the maximum value.
5, 6	00	$Izoi^i$ uses 8-bit integer
	01	$Izoi^i$ uses 16-bit integer
	10	$Izoi^i$ uses 32-bit integer
	11	$Izoi^i$ uses 64-bit integer
7, 8	00	$Izoi^i$ is described in one dimension
	10	$Izoi^i$ is described in two dimensions
	01	$Izoi^i$ is described in three dimensions
9	0	Offset with lengths mode is not used
	1	Offset with lengths mode is used: Specifies the initial offset with lengths of contiguous bytes that follows. The existence of this flag shall override the modes specified in bits 3 and 4.
		All other values are reserved

When TRLCP tags are used, their size is defined by P_{TRLCP} as specified in Table 4. In this case, bits 5 and 6 of the M_{ZOI} parameter are overridden.

The Offset with lengths mode can be used to efficiently represent a series of consecutive segments, for example, a series of consecutive byte ranges. The first value specifies the initial offset, the following values specify the lengths of each consecutive segment. If this field is used to represent n segments, then Nzoi should be set to n + 1.

5.8 Protection method template syntax (T)

5.8.1 General

Protection method templates contain parameters for specific JPSEC tools described in 5.6.1. For example, they are used in JPSEC normative tools described in 5.6.2. Also, they can be used in JPSEC non-normative tools described in 5.6.3. There are three types of protection method templates: decryption template, authentication template, and hash template. The template used by a JPSEC normative tool is specified by its ID as shown in Table 6 and again here in Table 21 with references to the appropriate clauses where they are defined.

As described in 5.6.2, the protection method template T together with a JPSEC tool's processing domain PD, granularity G, and value list V describe how a JPSEC tool is applied.

Table 21 – Template ID values (ID_T)

Values	Protection method template
0	Reserved
1	Decryption template. See 5.8.2.
2	Authentication template. See 5.8.3.
3	Hash template. See 5.8.4.
4	NULL tool
	All other values are reserved for ITU-T | ISO use

5.8.2 Decryption template ($T = T_{decry}$, if $T_{type} = 0$ and ID = 1)

The decryption template, T_{decry}, is used to communicate to the decryptor, how to decrypt the received codestream. Figure 23 shows the decryption template syntax. Table 22 shows the sizes and values of the symbols and parameters for the decryption template.

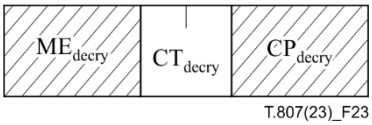


Figure 23 – Decryption template syntax

ME_{decry}: False marker emulation flag indicates whether a false marker emulation has occurred in the encrypted data. A false marker emulation can adversely affect compliance with JPEG 2000 Core coding system decoders. This field uses the FBAS structure.

CT_{decry}: Cipher type identification.

CP_{decry}: Cipher parameter.

Table 22 – Decryption template parameter values

Parameter	Size (bits)	Values
ME_{decry}	8 + 8 * n (FBAS)	Table 23
CT_{decry}	16	Table 24
CP_{decry}	Variable	If $CT_{decry} < 0x6000$, see 5.8.2.1. If $0x6000 \leq CT_{decry} < 0xC000$, see 5.8.2.2. If $CT_{decry} \geq 0xC000$, see 5.8.2.3.

Table 23 – Marker emulation flag values (ME_{decry})

Values	Method type
01xx xxxx	Encrypted data does not contain a false marker emulation
00xx xxxx	Otherwise
	All other values are reserved for ITU-T | ISO use

The default value of the marker emulation flag is 0. This flag can be set to 1 to indicate that the JPSEC encrypted data does not contain a false marker emulation. A JPSEC creator can choose to leave this flag at its default value of 0.

Table 24 – Cipher identifier values (CT_{decry})

Values	Cipher type
0 ... 0x5FFF	Block cipher (see Table 25)
0x6000 ... 0xBFFF	Stream cipher (see Table 26)
0xC000 ... 0xFFFF	Asymmetric cipher (see Table 27)

Table 25 – Block cipher identifier values (CT_{decry})

Values	Cipher type
0x0000	NULL (no encryption)
0x0001	AES (ISO/IEC 18033-3)
0x0002	TDEA (ISO/IEC 18033-3)
0x0003	MISTY1 (ISO/IEC 18033-3)
0x0004	Camellia (ISO/IEC 18033-3)
0x0005	CAST-128 (ISO/IEC 18033-3)
0x0006	SEED (ISO/IEC 18033-3)
	All other values are reserved for ITU-T | ISO use

Table 26 – Stream cipher identifier values (CT_{decry})

Values	Cipher type
0x6000	SNOW 2 (ISO/IEC 18033-4)
	All other values are reserved for ITU-T | ISO use

Table 27 – Asymmetric cipher identifier values (CT_{decry})

Values	Cipher type
0xC000	RSA-OAEP (ISO/IEC 18033-2)
	All other values are reserved for ITU-T | ISO use

5.8.2.1 Block cipher template (CP_{decry} for block ciphers)

The block cipher template is used to communicate to the block decryptor how to decrypt the received codestream. Figure 24 shows the block cipher mode, padding mode, block size and key information.

Some block cipher modes can use initialization vectors. For these modes, the tool's initialization vectors are specified using the tool's granularity field (G) described in 5.10 and Value list field (V) described in 5.11. Specifically, initialization vectors are only used for modes with ID M_{bc} > 0x80, for instance CBC, CFB, OFB, CTR. In the CTR case, it is not really an IV but a *counter*. The size of the initialization vector specified in the Value list V shall be set to the block size SIZ_{bc}.

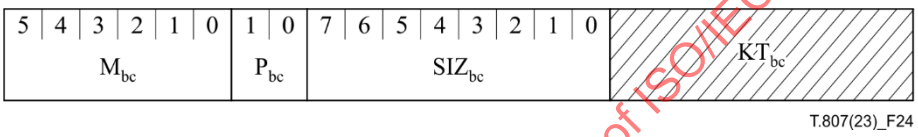


Figure 24 – Block cipher template syntax

M_{bc}: Block cipher mode. The first bit indicates the use of initialization vectors with this tool. If M_{bc} < 0x8, IVs are not used, otherwise one or more IV values are required for the mode.

P_{bc}: Padding mode.

SIZ_{bc}: Size of block in Bytes.

KT_{bc}: Key template (see 5.8.5). It holds information on the keys used by the block cipher.

Table 28 – Block cipher template values

Parameter	Size (bits)	Values
M_{bc}	6	Table 29
P_{bc}	2	Table 30
SIZ_{bc}	8	1 ... 256
KT_{bc}	Variable	See 5.8.5

Table 29 – Block cipher mode values (M_{bc})

Values	Mode type
0	Reserved
0x xxxx	Modes which are used without IV
1x xxxx	Modes which are used with an IV
x0 xxxx	P_{bc} bits are not padded, use 00
x1 xxxx	P_{bc} bits are padded with value defined in Table 30
0x 0001	ECB (ISO/IEC 10116)
1x 0010	CBC (ISO/IEC 10116)
1x 0011	CFB (ISO/IEC 10116)
1x 0100	OFB (ISO/IEC 10116)
1x 0101	CTR (ISO/IEC 18033-2)
	All other values are reserved for ITU-T | ISO use

NOTE 1 – Careful implementations are required for all modes, because improper implementations can lead to vulnerabilities. Note that even correct implementation of ECB has information leakage when identical blocks appear. Guidelines are contained in ISO/IEC 10116.

NOTE 2 – Values in Table 30 only apply when M_{bc} in Table 29 specifies that Bits are padded. When bits are not padded, P_{bc} shall be set to 00.

Table 30 – Padding mode for block cipher (P_{bc})

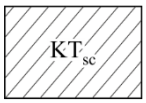
Values	Padding type
00	Ciphertext stealing (RFC 2040)
01	PKCS#7-padding (PKCS#7)
	All other values are reserved for ITU-T | ISO use

NOTE 3 – When using padding, careful system design should be used to avoid potential security vulnerabilities, such as chosen cipher attacks.

5.8.2.2 Stream cipher template (CP_{decry} for stream ciphers)

The stream cipher template is used to communicate to the stream decryptor how to decrypt the received codestream. Figure 25 shows the stream cipher template syntax. Table 31 shows the values of the stream cipher template.

The stream cipher's initialization vectors are specified using the tool's granularity field (G) described in 5.10 and Value list field (V) described in 5.11. The size of the initialization vector specified in the Value list V shall be set to the key size defined in the key information template KT_{sc}.



T.807(23)_F25

Figure 25 – Stream cipher template syntax

KT_{sc}: Key information template (see 5.8.5). It holds information on the keys used by the stream cipher.

Table 31 – Stream cipher template values

Parameter	Size (bits)	Values
KT_{sc}	Variable	See 5.8.5

5.8.2.3 Asymmetric cipher template (CP_{decry} for asymmetric ciphers)

The asymmetric cipher template is used to communicate to the asymmetric cipher decryptor, how to decrypt the received codestream. Figure 26 shows the asymmetric cipher template syntax. Table 32 shows the values of the asymmetric cipher template.

For tools that use the asymmetric cipher template, the tool's granularity field (G) specifies the granularity with which the cipher is applied. However, the Value list field (V) is not used to represent any values. Thus, the number of elements (N_v) in the Value list field shall be set to 0.

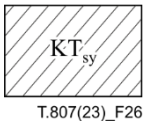


Figure 26 – Asymmetric cipher template syntax

KT_{sy}: Key information template (see 5.8.5). It holds information on the keys used by the asymmetric cipher.

Table 32 – Asymmetric cipher template values

Parameter	Size (bits)	Values
KT_{sy}	Variable	See 5.8.5

5.8.3 Authentication template (T = T_{auth}, if T_{type} = 0 and ID = 2)

The authentication template, T_{auth}, is used to communicate to the verifier, how to verify the authenticity of received codestream. There are three general classes of authentication methods: hash-based authentication, cipher-based authentication, and digital signatures. Both hash-based and cipher-based authentication methods are also generally referred to as message authentication codes (MACs), and their computed values which are used for authentication are generally referred to as MAC values. The authentication template syntax is shown in Figure 27, and Table 33 shows the sizes and values of the symbols and parameters for the authentication template.

In many security applications, authentication is the most important security service. Even when confidentiality is the targeted security service, it should be augmented by authentication to prevent attacks. In particular, it is recommended to authenticate parts of the SEC marker segment. In addition, the authentication shall be performed on both the authentication template parameters (T_{auth}) and the message to authenticate. Specifically, the zone of influence shall specify that both the content and the authentication template parameters (T_{auth}) are to be authenticated.

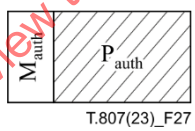


Figure 27 – Authentication template syntax

M_{auth}: Authentication method.

P_{auth}: Authentication parameters.

Table 33 – Authentication template parameter values

Parameter	Size (bits)	Values
M_{auth}	8	Table 34
P_{auth}	Variable	If M_{auth} = 0, see 5.8.3.1 If M_{auth} = 1, see 5.8.3.2 If M_{auth} =2, see 5.8.3.3

Table 34 – Authentication methods (M_{auth})

Values	Method
0	Hash-based MAC
1	Cipher-based MAC
2	Digital Signature
	All other values are reserved for ITU-T | ISO use

5.8.3.1 Hash-based authentication (P_{auth} for hash-based MAC)

The hash-based authentication MAC is used to communicate to the verifier how to verify the authenticity of the received codestream. Figure 28 shows the hash-based authentication template syntax and Table 35 shows the parameter values.

The MAC values are specified using the tool's granularity field (G) described in 5.10 and Value list field (V) described in 5.11. The size of the MAC value specified in the value list V shall be set to the MAC size defined by SIZ_{HMAC}.

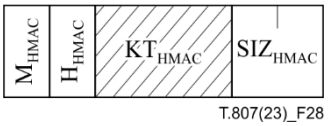


Figure 28 – Hash-based authentication template

M_{HMAC}: Hash-based authentication method identifier.

H_{HMAC}: Hash identifier.

KT_{HMAC}: Key template.

SIZ_{HMAC}: Size of MAC (bits).

Table 35 – Hash-based authentication template parameter values

Parameter	Size (bits)	Values
M_{HMAC}	8	Table 36
H_{HMAC}	8	Table 37
KT_{HMAC}	Variable	See 5.8.5
SIZ_{HMAC}	16	0 ... 65535

Table 36 – Hash-based authentication method identifier (M_{HMAC})

Values	Hash-based authentication method
0	Reserved
1	HMAC (ISO/IEC 9797-2)
	All other values are reserved for ITU-T | ISO use

Table 37 – Hash function identifier (H_{HMAC})

Values	Hash function
0	Reserved
1	SHA-1 (ISO/IEC 10118-3)
2	RIPEMD-128 (ISO/IEC 10118-3)
3	RIPEMD-160 (ISO/IEC 10118-3)
4	MASH-1 (ISO/IEC 10118-4)
5	MASH-2 (ISO/IEC 10118-4)
6	SHA-224 (ISO/IEC 10118-3)
7	SHA-256 (ISO/IEC 10118-3)
8	SHA-384 (ISO/IEC 10118-3)
9	SHA-512 (ISO/IEC 10118-3)
10	WHIRLPOOL (ISO/IEC 10118-3)
	All other values are reserved for ITU-T | ISO use

Note that if the SIZ_{HMAC} is less than the nominal size of the hash, then it is the truncated version corresponding to the first SIZ_{HMAC} bits of the hash.

5.8.3.2 Cipher-based authentication template (P_{auth} for cipher-based MAC)

The cipher-based authentication MAC is used to communicate to the verifier how to verify the authenticity of the received codestream. Figure 29 is its template and Table 38 shows the key size and keyed hash. An example cipher based authentication scheme is CBC-MAC. In these block-cipher techniques for authentication, the initialization vector is one blocksize in length and of value 0. The blocksize is the default for the block cipher. Note that if the SIZ_{CMAC} is less than the nominal size of the cipher-based authentication MAC, then it is the truncated version corresponding to the first SIZ_{CMAC} bits of the MAC.

Note that, if the number of data bits is not a multiple of the cipher block size, then the final input block will be a partial block of data, left justified, with zeroes appended to form a full cipher block. Also note that CBC-MAC shall only be applied to data with a fixed and known length.

The MAC values are specified using the tool's granularity field (G) described in 5.10 and value list field (V) described in 5.11. The size of the MAC value specified in the value list V shall be set to the MAC size defined by SIZ_{CMAC}.

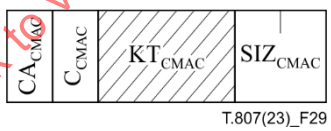


Figure 29 – Cipher-based authentication template syntax

CA_{CMAC}: Cipher-based authentication method.

C_{CMAC}: Block-cipher identifier value.

KT_{CMAC}: Key template.

SIZ_{CMAC}: Size of MAC (bits).

Table 38 – MAC template values

Parameter	Size (bits)	Values
CA_{CMAC}	8	Table 39
C_{CMAC}	8	Table 25
KT_{CMAC}	Variable	See 5.8.5
SIZ_{CMAC}	16	0 ... 65535

Table 39 – Cipher-based authentication method (C_{CMAC})

Values	Method
0	CBC-MAC MAC Algorithm 1 (ISO/IEC 9797-1)
1	CBC-MAC MAC Algorithm 2 (ISO/IEC 9797-1)
2	CBC-MAC MAC Algorithm 3 (ISO/IEC 9797-1)
3	CBC-MAC MAC Algorithm 4 (ISO/IEC 9797-1)
	All other values are reserved for ITU-T | ISO use

5.8.3.3 Digital signature template (P_{auth} for digital signatures)

The digital signature is used to communicate to the verifier how to verify the authenticity of received codestream, as well as verifying the identity of the sender for both identity and non-repudiation purposes. Figure 30 defines its template and Table 40 lists the values.

The digital signatures are specified using the tool's granularity field (G) described in 5.10 and value list field (V) described in 5.11. The size of the digital signatures value specified in the value list V shall be set to accommodate the size defined by SIZ_{DS}. Because the value list size is represented by bytes rather than bits, its size should be the minimum number of bytes that can accommodate SIZ_{DS}. Each value should be represented with the least significant bits, and the extra MSB bits shall be set to 0.

Figure 30 – Digital signature template syntax

M_{DSI}: Digital signature method.

H_{DS}: Hash function.

KT_{DS}: Key template (see 5.8.5). It holds all the information related to the public key or the certificate required to verify the digital signature.

SIZ_{DS}: Size of digital signature (bits).

Table 40 – Digital signature template values

Parameter	Size (bits)	Values
M_{DS}	8	Table 41
H_{DS}	8	Table 37
KT_{DS}	Variable	See 5.8.5
SIZ_{DS}	16	0 ... 65535

Table 41 – Digital signature methods (M_{DS})

Values	Method
1	RSA (ISO/IEC 14888-2)
2	Rabin (ISO/IEC 14888-2)
3	DSA (ISO/IEC 14888-3)
4	ECDSA (ISO/IEC 14888-3)
	All other values are reserved for ITU-T | ISO use

5.8.4 Hash template ($T = T_{hash}$, if $T_{type} = 0$ and ID = 3)

The hash template, T_{hash} as shown in Figure 31, is used to communicate the parameters used to compute the hash. Table 42 shows the sizes and values of the symbols and parameters for hash template.

Note that in contrast to the hash-based authentication template discussed in 5.8.3.1 which involves the use of a hash and a secret key, this hash template does not use a key. While this hash template can be used to detect an accidental error or accidental change to the data, it does not prevent malicious alteration of the data. In order to prevent malicious alteration

of the data an authentication template should be used, since the secret key used by the authentication template prevents the data from being altered without being discovered.

The hash values are specified using the tool's granularity field (G) described in 5.10 and value list field (V) described in 5.11. The size of the hash value specified in the value list V shall be set to the hash value size defined by SIZ_{hash}.

Figure 31 – Hash template syntax

H_{hash}: Hash function identifier.

SIZ_{hash}: Size of hash value (bytes).

Table 42 – Hash template parameter values

Parameter	Size (bits)	Values
H_{hash}	8	Table 37
SIZ_{hash}	8	0 ... 255

5.8.5 Key information template (KT)

The key information template is used to communicate key information. Figure 32 defines its template and Table 43 lists the values.

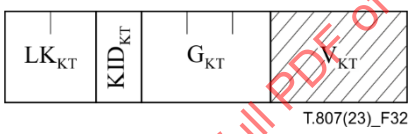


Figure 32 – Key information template syntax

LK_{KT}: Length of key in bits.

KID_{KT}: Key information identifier. It indicates the meaning of the values in the value list V_{KT}. In the decryption template, this value should be set to 2 (URI to retrieve the secret key). In the case of digital signature, the value of this field is free.

G_{KT}: Granularity field to represent the granularity with which the key information changes.

V_{KT}: Value list field to represent the changing list of key information.

Note that in the case of a secret key (decryption template), the public key and certificate have no meanings: the key template should hold some information on the location of the key (e.g., URI).

The key information can be represented with one or more values using the tool's granularity field (G_{KT}) described in 5.10 and value list field (V_{KT}) described in 5.11. The two fields (G_{KT} and V_{KT}) together determine how the key values in the value list (V_{KT}) are applied to the protected image data, as described in 5.10 and 5.11.

The key information in the value list can take one the forms specified in Table 44. If $KID_{KT} = 1$, then each value is specified with the ITU-T X.509 certificate template described in 5.8.5.1. If $KID_{KT} = 2$, then each value is specified with a URI for the certificate or secret key.

Table 43 – Key template values

Parameter	Size (bits)	Values
LK_{KT}	16	1 ... 65535
KID_{KT}	8	Table 44
G_{KT}	24	See 5.10
V_{KT}	Variable	See 5.11

Table 44 – Key information identifier values (KID_{KT})

Values	Key information identifier
0	Reserved
1	ITU-T X.509 certificate (ISO/IEC 9594-8)
2	URI for certificate or secret key
	All other values are reserved for ITU-T | ISO use

5.8.5.1 ITU-T X.509 certificate template

The ITU-T X.509 certificate template is shown in Figure 33 and Table 45 lists the values.

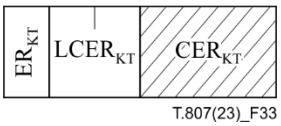


Figure 33 – ITU-T X.509 certificate syntax

ER_{KT}: Encoding rule for ITU-T X.509 certificate.

$LCER_{KT}$: Length of ITU-T X.509 certificate (CER_{KT}) in bytes.

CER_{KT}: ITU-T X.509 certificate.

Table 45 – ITU-T X.509 certificate values (KI_{KT} if KID_{KT} = 2)

Parameter	Size (bits)	Values
ER_{KT}	8	0 ... 255 (see Table 46)
$LCER_{KT}$	16	1 ... 65535
CER_{KT}	Variable	–

Table 46 – Encoding rule values (ER_{KT})

Values	Encoding rule identifier
0	Reserved
1	DER (RFC 3217)
2	BER (RFC 3394)
	All other values are reserved for ITU-T | ISO use

5.9 Processing domain syntax (PD)

The processing domain syntax, as shown in Figure 34, is used to indicate on which domain the JPSEC tool is applied. The possible domains include pixel domain, wavelet coefficient domain, quantized wavelet coefficient domain and codestream domain.

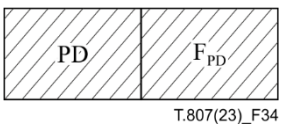


Figure 34 – Processing domain syntax

PD: Processing domain. This field uses the FBAS structure.

F_{PD}: Processing domain field to provide further detailed information about the processing domain. This field uses the FBAS structure.

Table 47 – Processing domain parameters

Parameter	Size (bits)	Values
PD	Variable (FBAS)	See Table 48
F_{PD}	Variable (FBAS)	In wavelet coefficient domain and quantized wavelet coefficient domain, see Table 49. In codestream domain, see Table 50.

Table 48 – Processing Domain (PD) parameter values

FBAS bit number	Values	Semantics
1	1	Pixel domain. Protection method is applied on image pixels.
	0	Otherwise
2	1	Wavelet coefficient domain. Protection method is applied on wavelet coefficients.
	0	Otherwise
3	1	Quantized wavelet coefficient domain: Protection method applied on quantized wavelet coefficient
	0	Otherwise
4	1	Codestream domain: Protection method is applied on codestream generated from arithmetic coder.
	0	Otherwise

Note that the field PD shall have one and only bit set to 1, because each JPSEC tool is applicable to one domain only.

In image pixel domain, wavelet coefficient domain and quantized wavelet coefficient domain, the two-dimensional data has to be transformed to one-dimensional in order to apply the security tools. This transformation shall be done by scanning the two-dimensional image data in the raster-scan order.

Table 49 – Processing domain field (F_{PD}) parameter values in wavelet coefficient domain and quantized wavelet coefficient domain

FBAS bit number	Value	Semantics
1	0	Protection method is applied on sign bit
	1	Protection method is applied on most significant bit

Table 50 – Processing domain field (F_{PD}) parameter values in codestream domain

FBAS bit number	Value	Semantics
1	0	Protection method is applied on both packet header and packet body
	1	Protection method is applied on packet body only

The field (F_{PD}) is used to provide further information on processing domain. With different value of PD, this field (F_{PD}) has different semantics. For instance, in wavelet coefficient domain and quantized wavelet coefficient domain, the first bit of F_{PD} is used to indicate whether the JPSEC tool is applied on the most significant bit. In codestream domain, the first bit of F_{PD} is used to indicate whether the JPSEC tool is applied on packet body only or both packet header and body; in the pixel domain, this field (F_{PD}) is reserved.

5.10 Granularity syntax (G)

Granularity is used to indicate the unit of protection for each protection method. Table 53 defines possible granularities. Figure 35 shows the granularity syntax.

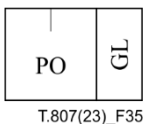


Figure 35 – Granularity syntax

PO: Processing order.

GL: Granularity level.

Table 51 – Granularity parameter values (G)

Parameter	Size (bits)	Values
PO	16	See Table 52
GL	8	See Table 53

Table 52 – Processing order values (PO)

Values MSB LSB	Processing order
0 000 000 000 000 000	Order specified by Zone of Influence image-related parameters
1 000 000 000 000 000	Order specified by Zone of Influence non-image-related bitstream parameters
1 000 000 000 000 001	Order specified by Zone of Influence non-image-related packet parameters
0 000 001 010 011 100	Tile-resolution-layer-component-precinct
0 000 011 100 001 010	Tile-component-precinct-resolution-layer
0 000 010 001 011 100	Tile-layer-resolution-component-precinct
0 000 100 011 001 010	Tile-precinct-component-resolution-layer
0 000 001 100 011 100	Tile-resolution-precinct-component-layer
	All other values are reserved

Table 53 – Granularity level values (GL)

Values MSB LSB	Granularity
0000 0000	Tile
0000 0001	Tile-part
0000 0010	Component
0000 0011	Resolution level
0000 0100	Layer
0000 0101	Precinct
0000 0110	Packet
0000 0111	Sub-band
0000 1000	Code-block
0000 1001	Total area identified in ZOI
1000 0000	Item identified in non-image-related ZOI
1000 0001	Zone identified in non-image-related ZOI
	All other values are reserved

In order to process the entire zone specified by ZOI, the granularity level should be "zone identified in ZOI".

5.11 Value list syntax (V)

The Value list field, as shown in Figure 36, is used to specify values that change as the tool is applied and the granularity with which it changes. This is used to signal changing values such as keys, initialization vectors, MAC values, digital signatures, and hash values. The Value List field first specifies the number of values in the list and the size of each value. It then lists the values themselves.

As discussed in 5.6.2, for JPSEC normative tools the Value list field represents a different parameter for each template. For the decryption template, it represents the initialization vectors IV_{bc} or IV_{sc} depending on whether a block cipher or stream cipher is used. For the authentication template, it represents the MAC value VAL_{MAC} for hash-based and cipher-based authentication. For the digital signature template, it represents the digital signature SIG_{DS}. For the hash template, it represents the hash value HV_{hash}. Some usages of the templates do not require values to be specified, e.g., not all decryption modes use initialization vectors. In these cases, the Value list field should set N_v and S_v equal to zero so that

the value list VL has no elements. If only a single value needs to be specified, e.g., if a single key is used throughout the image, then N_V will be set to one so that a single value is contained in the value list.

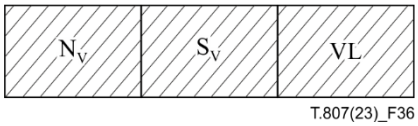


Figure 36 – Value list field syntax

N_V: Number of values in the value list VL, If $N_V = 0$, then the field terminates. This field uses RBAS structure.

S_V: Size of each value in the value list VL in bytes. This field uses RBAS structure.

VL: List of values.

Table 54 – Value list field (V) parameter values

Parameter	Size (bits)	Values
N_V	16 + 8 * n (RBAS)	0 ... $(2^{15+7*n} – 1)$
S_V	8 + 8 * n (RBAS)	0 ... $(2^{7+7*n} – 1)$
VL	0, if NV = 0 N_V * S_V, otherwise	N/A Determined by template

5.12 Relationships among ZOI, Granularity (G) and Value List (VL)

The ZOI, PO and GL are used together to ensure the unique behaviour of the applied JPSEC tool(s), regardless of the progress order of the JPEG 2000 codestream. In other words, the resulted signature, MAC values and encrypted codestream are independent of the progressive order of the JPEG 2000 codestream. The Zone of Influence (ZOI) specifies, in its entirety, the part of the JPEG 2000 codestream to be protected by the JPSEC tool; The Processing Order (PO), on the other hand, specifies the order in which the JPSEC tool processes the codestream; the Granularity Level (GL) specifies the protection units containing contiguous byte sequence in the re-ordered codestream. Finally, each protection unit corresponds to a value in the Value List (VL), in the order they appear in the re-ordered codestream. The relationship can be illustrated by one example, where the JPEG 2000 codestream has 1 tile, 3 resolution levels and 3 layers, and the number of components and precincts are not important. The progressive order is RLCP in original JPEG 2000 codestream, the Zone of Influence is resolution 0 and 1, and the Processing Order (PO) is TRLCP. Figure 37 and Figure 38 illustrate the re-ordering of the codestream and the mapping from each protection unit to the Value List (VL), when the Granularity Level (GL) is resolution and layer, respectively.

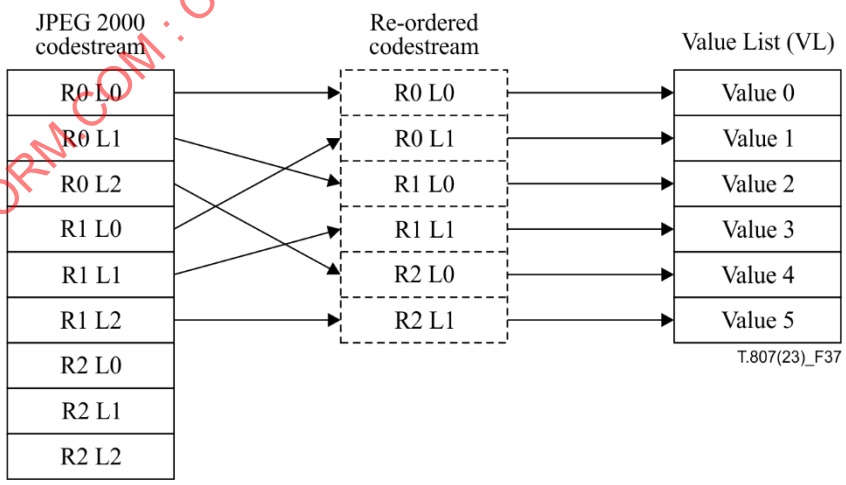


Figure 37 – Granularity Level (GL) is resolution

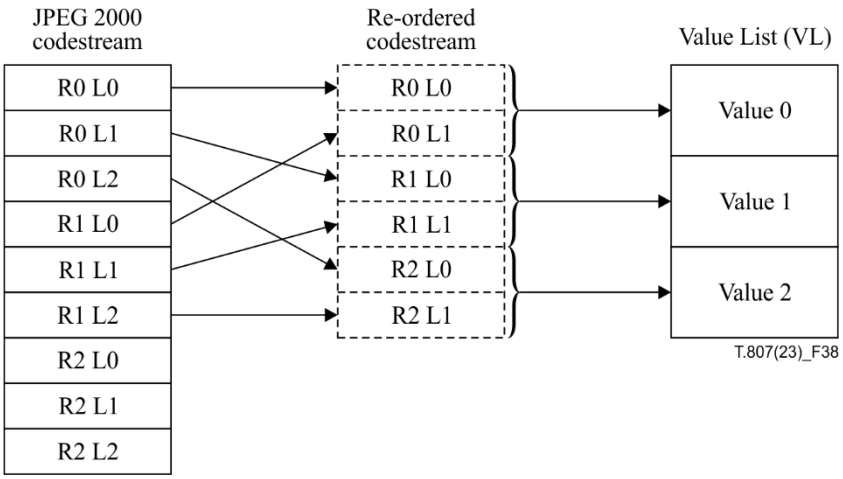


Figure 38 – Granularity Level (GL) is layer

NOTE – The re-ordered codestream is only used to generate the values in Value List (VL). The final JPSEC codestream will have the same progressive order as the original JPEG 2000 codestream.

5.13 In-codestream security marker (INSEC)

The in-codestream security marker (INSEC) provides an additional means to transmit security information. It is optional and is used in conjunction with the SEC security marker. Specifically, it is used in conjunction with a JPSEC non-normative tool.

More precisely, the SEC marker is present in the main header and gives overall information about the JPSEC tools applied to protect the image. The INSEC marker is present in the bitstream data itself and gives additional or alternative parameters for the JPSEC non-normative tool identified by the tool instance index parameter. Therefore, the tool instance index in the INSEC marker shall correspond to one of the tool instance index in the main header.

The INSEC marker segment can be placed in the bitstream data. It uses the fact that the arithmetic decoder in JPEG 2000 stops reading bytes from the bitstream when it encounters a termination marker (i.e., two bytes with a value greater than 0xFF8F).

The information carried in the INSEC marker segment is relevant for the preceding or following secured codeblock(s), until another INSEC marker is found.

Note that inclusion of INSEC markers results in a file that does not necessarily comply with the JPEG 2000 Core coding system. Note that some decoders can have difficulty in handling a marker in the middle of a packet. Insertion anywhere inside a packet will invalidate the length of the packet as indicated in the packet header. Also, there can be issues with encryption and INSEC markers due to:

a) lack of marker emulation restrictions on the encryption; and/or

b) inability to locate the marker itself in the presence of encryption.

The syntax of the INSEC marker is defined in Figure 39.

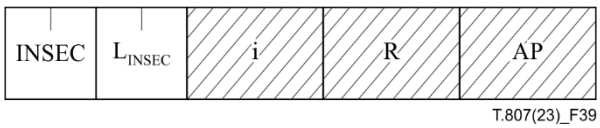


Figure 39 – In-codestream security marker syntax

INSEC: Marker code. Table 55 shows the sizes and values of the symbols and parameters for in-codestream security marker segment.

L_{INSEC}: Length of marker segment in bytes (not including the marker). Note that the INSEC marker segment should be byte-aligned.

i: Tool instance index corresponding to one of the tool instance index parameters in the SEC marker segment and therefore identifying the instance of the JPSEC tool this INSEC marker is referring to. This field uses the RBAS structure.

R: Relevance zone for the INSEC information. This field uses the FBAS structure.

AP: Additional or alternative parameters for protection method. The encoder should always make sure that the encoder does not emulate a marker in this parameter.

Table 55 – In-codestream security parameter values (INSEC)

Parameter	Size (bits)	Values
INSEC	16	0xFF94
L_{INSEC}	16	2 ... (2^{16} – 1)
i	8 + 8 * n (RBAS)	0 ... (2^{7+7*n} – 1)
R	Variable (FBAS)	See Table 56
AP	Variable	Defined by user defined application.

Table 56 – Relevance zone values (R)

FBAS bit number	Values	Relevance zone
0	0	Preceding code-blocks
	1	Following code-blocks

Because INSEC is used in conjunction with JPSEC non-normative tools, the format of the additional or alternative parameters is defined by the tool itself which is identified by the tool ID. Specifically, JPSEC non-normative tools are defined by private JPSEC applications. Thus, the definition of these tools should include the INSEC usage if it is allowed.

6 Normative-syntax usage examples (informative)

6.1 ZOI examples

6.1.1 to 6.16 contain examples that show how the Zone of Influence syntax can be used.

In the examples, the superscripts used in Pzoi, Mzoi, and Izoi correspond to the index of the image-related and non-image-related items signalled by the BAS structure in DCzoi in the order they appear within the DCzoi.

6.1.1 Example 1

This clause shows the example that resolution levels more than 3 in the image region whose upper-left corner is (100, 120) and lower-right (180, 210) are influenced. In this example, 9 bytes are necessary.

Table 57 – ZOI in example 1

Parameter			Size (bits)	Value (in order)	Derived meaning
NZzoi			8 (RBAS)	1	Number of Zones is one
$Zone^0$	DCzoi		1	0_b	The byte aligned segment does not follow
			1	0_b	Image related description class
			6	101000_b	Image regions and resolution levels are specified in order
	$Pzoi^1$	$Mzoi^1$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	00_b	Rectangle mode
			2	00_b	Izoi uses 8-bit integer
			1	1_b	Izoi is described in two dimensions
		$Izoi^1$	8	$0110\ 0100_b$	Xul is 100
			8	$0111\ 1000_b$	Yul is 120
			8	$1011\ 0100_b$	Xlr is 180
			8	$1101\ 0010_b$	Ylr is 210
	$Pzoi^3$	$Mzoi^3$	1	0_b	The byte aligned segment does not follow

Table 57 – ZOI in example 1

Parameter			Size (bits)	Value (in order)	Derived meaning
			1	1_b	The complement of the specified zones is influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	11_b	Max mode
			2	00_b	Izoi uses 8-bit integer
			1	0_b	Izoi is described in one dimension
		Izoi[3]	8	$0000\ 0010_b$	Resolution levels ≤ 2 are specified. (i.e., Resolution levels > 3 are specified with Max mode and complement switch.)

6.1.2 Example 2

This clause shows the example that the code-blocks whose upper-left corner's index is 5 and lower-right corner's index is 10 in the sub-band 1, in the resolution level 0 are influenced. In this example, 10 bytes are necessary.

Table 58 – ZOI in example 2

Parameter			Size (bits)	Value (in order)	Derived meaning
NZzoi			8 (RBAS)	1	Number of Zones is one
Zone[0]	DCzoi[1]		1	1_b	The byte aligned segment follows
			1	0_b	Image related description class
			6	001000_b	Resolution levels are specified
	DCzoi[2]		1	0_b	The byte aligned segment does not follow
			1	0_b	Image related description class
			6	001100_b	Sub-bands and code-blocks are specified
	Pzoi[3]	Mzoi[3]	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	10_b	Index mode
			2	00_b	Izoi uses 8-bit integer
			1	0_b	Izoi is described in one dimension
		Izoi[3]	8	$0000\ 0000_b$	Resolution level index is 0
	Pzoi[9]	Mzoi[8]	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	10_b	Index mode
			2	00_b	Izoi uses 8-bit integer
			1	0_b	Izoi is described in one dimension
		Izoi[8]	8	$0000\ 0001_b$	Sub-band 1 is specified
	Pzoi[10]	Mzoi[9]	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	00_b	Rectangle mode
			2	00_b	Izoi uses 8-bit integer
			1	0_b	Izoi is described in one dimension
		Izoi[9]	8	$0000\ 0101_b$	Code-block index for the upper-left corner is 5
			8	$0000\ 1010_b$	Code-block index for the lower-right corner is 10

6.1.3 Example 3

This clause shows the example that the data segments from bytes 10 to 100 and from bytes 10000 to 12000 are influenced. In this example, 12 bytes are necessary.

Table 59 – ZOI in example 3

Parameter			Size (bits)	Value (in order)	Derived meaning
NZzoi			8 (RBAS)	1	Number of Zones is one
$Zone^0$	DCzoi		1	0_b	The byte aligned segment does not follow
			1	1_b	Non-image related description class
			6	010000_b	Byte ranges after the SOD marker are specified
	$Pzoi^2$	$Mzoi^2$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	1_b	Multiple items are specified
			2	01_b	Range mode
			2	01_b	Izoi uses 16-bit integer
			1	0_b	Izoi is described in one dimension
		$Nzoi^2$	8	$0000\ 0010_b$	Number of data segments is 2
		$Izoi^{21}$	16	$0000\ 0000_b\ 0000\ 1010_b$	Starting byte location is 10th (bytes)
			16	$0000\ 0000_b\ 0110\ 0100_b$	Ending byte location is 100th (bytes)
		$Izoi^{21}$	16	$0010\ 0111_b\ 0001\ 0000_b$	Starting byte location is 10000th (bytes)
			16	$0010\ 1110_b\ 1110\ 0000_b$	Ending byte location is 12000th (bytes)

6.1.4 Example 4

This clause shows the example that resolution level 0 is influenced and that the byte segments 10 through 100 correspond to the data for resolution level 0. In this example, 10 bytes are necessary.

Table 60 – ZOI in example 4

Parameter			Size (bits)	Value (in order)	Derived meaning
NZzoi			8 (RBAS)	1	Number of Zones is one
$Zone^0$	$DCzoi^1$		1	1_b	The byte aligned segment follows
			1	0_b	Image related description class
			6	001000_b	Resolution levels are specified in order
	$DCzoi^2$		1	0_b	The byte aligned segment does not follow
			1	1_b	Non-image related description class
			6	010000_b	Byte ranges are specified
	$Pzoi^1$	$Mzoi^1$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	10_b	Index mode
			2	00_b	Izoi uses 8-bit integer
			1	0_b	Izoi is described in one dimension
		$Izoi^1$	8	$0000\ 0000_b$	Resolution level is 0
$Zone^0$	$Pzoi^2$	$Mzoi^2$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_b	Single items specified
			2	01_b	Range mode
			2	01_b	Izoi uses 16-bit integer
			1	0_b	Izoi is described in one dimension
		$Izoi^1$	16	$0000\ 0000\ 0000\ 1010_b$	Starting byte location is 10th (bytes)
			16	$0000\ 0000\ 0110\ 0100_b$	Ending byte location is 100th (bytes)

6.1.5 Example 5

This clause shows the example that resolution levels more than 3 in the tiles whose upper-left tile index is 0 and lower-right tile index is 5, and layers equal to or less than 5 in the tiles whose upper-left tile index is 10 and lower-right tile index is 15 are influenced. In this example, 13 bytes are necessary.

Table 61 – ZOI in example 5

Parameter			Size (bits)	Value (in order)	Derived meaning
NZzoi			8 (RBAS)	2	Number of Zones is two
$Zone^0$	DCzoi		1	0_b	The byte aligned segment does not follow
			1	0_b	Image related description class
			6	$01\ 1000_b$	Tiles and resolution levels are specified in order
	$Pzoi^2$	$Mzoi^2$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	00_b	Rectangle mode
			2	00_b	Izoi uses 8-bit integer
			1	0_b	Izoi is described in one dimension
		$Izoi^2$	8	$0000\ 0000_b$	Upper-left tile index is 0
			8	$0000\ 0101_b$	Lower-right tile index is 5
	$Pzoi^3$	$Mzoi^3$	1	0_b	The byte aligned segment does not follow
			1	1_b	The complement of the specified zones is influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	11_b	Max mode
			2	00_b	Izoi uses 8-bit integer
			1	0_b	Izoi is described in one dimension
		$Izoi^3$	8	$0000\ 0010_b$	Resolution levels ≤ 2 are specified. (i.e., Resolution levels > 3 are specified with Max mode and complement switch.)
$Zone^1$	DCzoi		1	0_b	The byte aligned segment does not follow
			1	0_b	Image related description class
			6	010100_b	Tiles and layers are specified in order
	$Pzoi^2$	$Mzoi^2$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	00_b	Rectangle mode
			2	00_b	Izoi uses 8-bit integer
			1	0_b	Izoi is described in one dimension
		$Izoi^2$	8	$0000\ 1010_b$	Upper-left tile index is 10
			8	$0000\ 1111_b$	Lower-right tile index is 15
	$Pzoi^4$	$Mzoi^4$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	11_b	Max mode
			2	00_b	Izoi uses 8-bit integer
			1	0_b	Izoi is described in one dimension
		$Izoi^4$	8	$0000\ 0101_b$	layers ≤ 5 are specified with Max mode

6.1.6 Example 6

This clause shows the example that the header segment from bytes 10 to 100 is influenced. In this example, 8 bytes are necessary.

Table 62 – ZOI in example 6

Parameter			Size (bits)	Value (in order)	Derived meaning
NZzoi			8 (RBAS)	1	Number of Zones is one
$Zone^0$	DCzoi		1	0_b	The byte aligned segment does not follow
			1	1_b	Non-image related description class
			6	001000_b	Byte ranges after the SEC marker are specified
	$Pzoi^3$	$Mzoi^3$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	01_b	Range mode
			2	01_b	Izoi uses 16-bit integer
			1	0_b	Izoi is described in one dimension
		$Izoi^3$	16	$0000\ 0000\ 0000\ 1010_b$	Starting byte location is 10th (bytes)
			16	$0000\ 0000\ 0110\ 0100_b$	Ending byte location is 100th (bytes)

6.2 Key information template examples

6.2.1 Example 1

Table 63 shows the example that a single secret key (128 bits) is used to decrypt a codestream, where the secret key is identified using URI and retrieved from the key server based on URI in the decryption stage.

Table 63 – Key information in example 1

Parameter		Size (bits)	Value	Derived meaning
LK_{KT}		16	128	Length of key is 128 bits
KID_{KT}		8	2	URI for secret key is identified
G_{KT}	PO	16	$000\ 001\ 010\ 011\ 100\ 0_b$	Processing order is tile-resolution-layer-component-precinct
	GL	8	$0000\ 1001_b$	Unit of protection is the total area identified in ZOI
V_{KT}	N_V	16 (RBAS)	1	Number of values in the value list V is 1
	S_V	8 (RBAS)	19	Length of key information is 19 bytes
	V1	152	https://server/file	Secret key can be retrieved from https://server/file

6.2.2 Example 2

Table 64 shows the example that an ITU-T X.509 certificate is used to authenticate a codestream, where the ITU-T X.509 certificate is embedded into KI_{KT} with encoding method DER.

Table 64 – Key information in example 2

Parameter			Size (bits)	Value	Derived meaning
LK_{KT}			16	1024	Length of key is 1024 bits
KID_{KT}			8	2	ITU-T X.509 certificate is identified
G_{KT}	PO		16	$000\ 001\ 010\ 011\ 100\ 0_b$	Processing order is tile-resolution-layer-component-precinct
	GL		8	$0000\ 1001_b$	Unit of protection is total area identified in ZOI
V_{KT}	N_V		16 (RBAS)	1	Number of values in the value list V is 1
	S_V		8 (RBAS)	Variable	Length of ITU-T X.509 certificate
	V1	ER_{KT}	8	1	ITU-T X.509 certificate is encoded with encoding method DER
		$LCER_{KT}$	16	Variable	Length of CER_{KT}
		CER_{KT}	Variable	Certificate value	Certificate with 1024-bit public key is embedded

6.2.3 Example 3

Table 65 shows that a single public key is used to authenticate a codestream, where the public key is embedded into KI_{KT}.

Table 65 – Key information in example 3

Parameter		Size (bits)	Value	Derived meaning
LK_{KT}		16	1024	Length of key is 1024 bits
KID_{KT}		8	1	Public key is identified
G_{KT}	PO	16	000 001 010 011 100 0_b	Processing order is tile-resolution-layer-component-precinct
	GL	8	0000 1001_b	Unit of protection is total area identified in ZOI
V_{KT}	N_V	16 (RBAS)	1	Number of values in the value list V is 1
	S_V	8 (RBAS)	256	Length of public key is 256 bytes
	V1	2048	Public key value	Public key is embedded

6.2.4 Example 4

Table 66 shows that multiple secret keys are used to decrypt a codestream, where different secret keys are used for different layers.

Table 66 – Key information in example 4

Parameter		Size (bits)	Value	Derived meaning
LK_{KT}		16	128	Length of key is 128 bits
KID_{KT}		8	3	URI for secret key is identified
G_{KT}	PO	16	000 001 010 011 1000_b	Processing order is tile-resolution-layer-component-precinct
	GL	8	0000 0100_b	Unit of protection is layer
V_{KT}	N_V	16 (RBAS)	3	Number of values in the value list V is 3
	S_V	8 (RBAS)	16	Length of each Vn is 16 bytes
	V1	128	https://server/1	Secret key for the 1st layer can be retrieved from https://server/1
	V2	128	https://server/2	Secret key for the 2nd layer can be retrieved from https://server/2
	V3	128	https://server/3	Secret key for the 3rd layer can be retrieved from https://server/3
	V4	128	https://server/4	Secret key for the 4th layer can be retrieved from https://server/4

6.3 JPSEC normative tool examples

The examples in 6.3.1 and 6.3.2 describe how the ZOI and key templates can be used to perform basic security services such as encryption and authentication on a JPEG 2000 coded image.

6.3.1 Example 1

An image is coded with JPEG 2000 and has three resolutions. In this example the first resolution is not encrypted in order to provide preview capability, and the second and third resolutions are encrypted with keys k1 and k2, respectively. The input image in this case is coded in RLCP progression order, and has 1 tile, 3 resolutions, 3 layers, Nc components, and Np precincts (the number of components and precincts is not significant in this specific example). Encryption is performed using AES in CBC mode without padding (using cipher-text stealing), using key k0 to encrypt resolution 1 and using key k2 to encrypt resolution 2, and resolution 0 is left unencrypted.

JPSEC signals how a JPSEC consumer should decrypt the JPSEC codestream. First, the tool template ID for the decryption template is signalled. Two ZOIs are specified for resolution 1 and its corresponding byte range B0-B1, and for resolution 2 and its corresponding byte range B2-B3. The decryption template parameters identify that AES encryption is applied without padding (using cipher-text stealing). The keying information and the fact that different keys are applied to different resolutions are signalled with the key information parameters. Specifically, the key granularity is specified as resolution so each resolution has a different key, where the processing order is signalled as TRLCP. The key information for each resolution is contained in the value list of keys. The encryption is performed on the codestream, encrypting both the packet headers and packet bodies. The encryption granularity is resolution, where the processing is performed in TRLCP ordering which is the same ordering as the original codestream. Since the two resolutions are encrypted separately, two initialization vectors (IVs) are required, and these are contained in the value list.

Note that packet's cipher text results are specified by the processing order and therefore are independent of input codestream's progression order; however, the placement of the encrypted packets in the output codestream follow the ordering of the input codestream packets.

Table 67 – SEC marker segment for example 1

Parameter			Size (bits)	Values	Meaning
SEC			16	0xFF65	SEC marker
L_{SEC}			16 (RBAS)	0x82	Length of SEC marker segment is 130 bytes
Z_{SEC}			8 (RBAS)	0	Index of this SEC marker segment
P_{SEC}	F_{PSEC}		1	0_b	The FBAS structure does not follow
		F_{INSEC}	1	0_b	INSEC is not used
		$F_{multiSEC}$	1	0_b	One SEC marker segment is used
		F_{mod}	2	00_b	Original JPEG 2000 data was modified
		F_{TRLCP}	1	0_b	TRLCP tag usage is not defined in P_{SEC}
		F_{TRLCP}	3	000_b	
	N_{tools}		8 (RBAS)	0000001_b	Number of security tool is one
	I_{max}		8 (RBAS)	0000000_b	Maximum tool instance index is zero
t			8 (FBAS)	0	JPSEC normative tool
i			8 (RBAS)	0	Tool instance index
ID_T			8	1	Decryption template
L_{zoi}			16 (RBAS)	0x17	Length of ZOI is 23 bytes
ZOI			184	See Table 68	Zone of Influence for this tool
L_{PID}			16 (RBAS)	0x5e	Length of P_{ID} is 94 bytes
P_{ID}			752	See Table 69	Parameters for this technology

Table 68 – ZOI example

Parameter			Size (bits)	Value (in order)	Derived meaning
NZzoi			8 (RBAS)	2	Number of Zones is one
$Zone^0$	DC_{ZOI}^1		1	1_b	The byte aligned segment follows
			1	0_b	Image related description class
			6	001000_b	Resolution is specified
	DCzoi[2]		1	0_b	The byte aligned segment does not follow
			1	1_b	Non-image related description class
			6	010000_b	Byte ranges after the SOD marker are specified
	$Pzoi^{0,1}$	$Mzoi^1$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	10_b	Index mode
			2	00_b	Izoi uses 8-bit integer
			1	0_b	Izoi is described in one dimension
		I_{ZOI}	8	$0000\ 0001_b$	Resolution 1 is specified
	$Pzoi^{0,2}$	$Mzoi^2$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	01_b	Range mode
			2	01_b	Izoi uses 16-bit integer
			1	0_b	Izoi is described in one dimension
		$Izoi^{21}$	16	0x31CC	Starting byte location is 12748 (bytes). (B0)
			16	0xA3E8	Ending byte location is 41960 (bytes). (B1)

Table 68 – ZOI example

Parameter			Size (bits)	Value (in order)	Derived meaning
Zone[1]	DC_{ZOI}[1]		1	1_b	The byte aligned segment follows
			1	0_b	Image related description class
			6	001000_b	Resolution is specified
	DCzoi[2]		1	0_b	The byte aligned segment does not follow
			1	1_b	Non-image related description class
			6	010000_b	Byte ranges after the SOD marker are specified
	$Pzoi^{0,1}$	$Mzoi^1$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	10_b	Index mode
			2	00_b	Izoi uses 8-bit integer
			1	0_b	Izoi is described in one dimension
		I_{ZOI}[1]	8	$0000\ 0010_b$	Resolution 2 is specified
	$Pzoi^{0,2}$	$Mzoi^2$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_{b2}	Single item is specified
			2	01_b	Range mode
			2	10_b	Izoi uses 32-bit integer
			1	0_b	Izoi is described in one dimension
		Izoi[2]	32	0xA3EE	Starting byte location is 41966 (bytes). (B2)
			32	0x21101	Ending byte location is 135425 (bytes). (B3)

Table 69 – P_{ID} example

Parameter		Size (bits)	Values	Meaning
T_{ID}		432	See Table 70	Decryption templates
PD		8 (FBAS)	0_b	The byte aligned segment does not follow
			0_b	Pixel domain is not used
			0_b	Wavelet coefficient domain is not used
			0_b	Quantized wavelet coefficient domain is not used
			1_b	Codestream domain is used
			000_b	Reserved for ITU-T | ISO use
F_{PD}		8 (FBAS)	0_b	The FBAS byte does not follow
			1_b	Only packet body is encrypted
			000000_b	Reserved for ITU-T | ISO use
G	PO	16	$000\ 001\ 010\ 011\ 100\ 0_b$	Processing order is tile-resolution-layer-component-precinct
	GL	8	$0000\ 0011_b$	Unit of protection is resolution level
V	N_V	16 (RBAS)	2	Number of values in the value list V is 2
	S_V	8 (RBAS)	16	Length of each Vn is 16 bytes
	V1	128	IV0	Initialization vector value for R1
	V2	128	IV1	Initialization vector value for R2

Table 70 – Decryption template example

Parameter		Size	Value (in order)	Derived meaning
ME_{decry}		8	0	Marker emulation has occurred
CT_{decry}		16	0001_b	Block cipher (AES)
CP_{decry}	M_{bc}	6	100000_b	CBC mode. Bits are not padded
	P_{bc}	2	00_b	Ciphertext stealing
	SIZ_{bc}	8	16	Block size (16 bytes, 128 bits)
	KT_{bc}	392	See Table 71	Key template

Table 71 – Key template example

Parameter		Size (bits)	Value	Derived meaning
LK_{KT}		16	128	Length of key is 128 bits
KID_{KT}		8	2	URI for secret key
G_{KT}	PO	16	$0\ 000\ 001\ 010\ 011\ 100_b$	Processing order is tile-resolution-layer-component-precinct
	GL	8	$0000\ 0011_b$	Unit of protection is resolution level
V_{KT}	N_V	32 (RBAS)	2	Number of values in the value list V is 2
	S_V	8 (RBAS)	19	Length of each Vn is 19 bytes
	V1	152	https://server/key1	Secret key for resolution level 1 can be retrieved from https://server/key1
	V2	152	https://server/key2	Secret key for resolution level 2 can be retrieved from https://server/key2

6.3.2 Example 2

In this case, authentication is applied to the same JPEG 2000 coded image as above. In this example all three resolutions and three layers per resolution are authenticated, where the authentication of each resolution uses a different key. Since there are three resolutions there are three keys, and since there are three layers per resolution there will be three MAC values per resolution. Thus, there will be a total of nine MAC values for the entire JPSEC image. Specifically,

- Resolution 0 has MAC values M0, M1, M2 (one for each layer) using key0
- Resolution 1 has MAC values M3, M4, M5 (one for each layer) using key1
- Resolution 2 has MAC values M6, M7, M8 (one for each layer) using key2

This example illustrates how authentication can be signalled as well as the flexibility provided by the ZOI and granularity tools. As in the prior example, the input image is coded in RLCP progression order, and has 1 tile, 3 resolutions, 3 layers, Nc components, and Np precincts (the number of components and precincts is not important in this specific example). Authentication is performed using HMAC with SHA-1.

JPSEC signals how a JPSEC consumer can verify or authenticate the JPSEC protected content. First, the tool template ID for the authentication template is signalled. Then the ZOI is used to signal that there are three resolutions and the associated byte ranges for each resolution. The authentication template parameters signal that HMAC is applied using SHA-1. The key information template provides information about the keys including that the key granularity is resolution and supplying the information for each of the three keys in the value list for the keys. The processing domain for authentication is specified as the codestream including packet headers. The tool granularity for authentication is specified as the layer, therefore there are 3 MACs for each resolution, for a total of nine MAC values. The value list contains the nine MAC values. The processing order for the above was identified as TRLCP, which is the same as the original codestream order.

Note that the use of processing order in the granularity field ensures that the same MAC values would result independent of the codestream's progression order.

Note that while this example demonstrated the use of MACs, the same approach can be used to signal the use of multiple digital signatures.

Table 72 – The SEC marker segment

Parameter			Size (bits)	Value (in order)	Derived meaning
SEC			16	0xFF65	SEC Marker
L_{SEC}			16	0x0099	Length of SEC marker segment
Z_{SEC}			8 (RBAS)	0	Index of this SEC marker segment
P_{SEC}	F_{PSEC}		1	0_b	The FBAS structure does not follow
		F_{INSEC}	1	0_b	INSEC marker segment is not used
		$F_{multiSEC}$	1	0_b	Only one SEC marker segment in this codestream
		F_{mod}	1	0_b	Original JPEG 2000 data was not modified
		F_{TRLCP}	1	0_b	The TRLCP tag is not used
		Padding	3	000_b	The TRLCP tag is not used
	N_{tools}		7	1	Only one tool is used in this codestream
	I_{max}		7	0	The maximum tool instance index is 0
$Tool^0$	t		8 (FBAS)	0	JPSEC Normative tool
	i		8 (RBAS)	0	Tool instance index
	ID_T		8	2	This normative tool uses an authentication template
	L_{ZOI}		16 (RBAS)	0x20	Length of ZOI is 32 bytes
	ZOI		256	Table 73	The covered zone of the image
	L_{PID}		16 (RBAS)	0x6c	Length of P_{ID} is 108 bytes
	P_{ID}		928	Table 74	Parameters for JPSEC tool

Table 73 – ZOI signalling

Parameter			Size (bits)	Value (in order)	Derived meaning
NZzoi			8 (RBAS)	1	Number of Zones is 1
$Zone^0$	DC_{zoi}^1		1	1_b	The byte aligned segment follows
			1	0_b	Image related description class
			6	001000_b	Resolution levels are specified in order
	DC_{zoi}^2		1	0_b	The byte aligned segment does not follow
			1	1_b	Non-image related description class
			6	010000_b	Byte ranges are specified
	$Pzoi^{0,1}$	$Mzoi^1$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	01_b	Range mode
			2	00_b	Izoi uses 8-bit integer
			1	0_b	Izoi is described in one dimension
		$Izoi^1$	8	0	The beginning of the range is 0
			8	2	The end of the range is 2
$Zone^0$	$Pzoi^{0,2}$	$Mzoi^2$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	1_b	Multiple items specified
			2	01_b	Range mode
			2	10_b	Izoi uses 32-bit integer
			1	0_b	Izoi is described in one dimension
		N_{ZOI}	8 (RBAS)	3	Number of I_{ZOI} is 3
		$Izoi^1$	32	104	Starting byte location is 104 (bytes)
			32	12762	Ending byte location is 12762 (bytes)

Table 73 – ZOI signalling

Parameter			Size (bits)	Value (in order)	Derived meaning
		I_{ZOI}^2	32	12768	Starting byte location is 12768 (bytes)
			32	41980	Ending byte location is 41980 (bytes)
		I_{ZOI}^3	32	41986	Starting byte location is 41986 (bytes)
			32	135445	Ending byte location is 135445 (bytes)

Table 74 – P_{ID} signalling parameters

Parameter					Size (bits)	Value (in order)	Derived meaning
T_{auth}	M_{auth}				8	0	Authentication methods: Hash-based authentication
	P_{auth}	M_{HMAC}			8	1	HMAC is used for authentication
		H_{HMAC}			8	1	SHA-1 is used for hashing
		KT_{HMAC}	LK_{KT}		16	128	Length of the key in bits
			KID_{KT}		8	3	KI_{KT} contains the URI for the private key
			G_{KT}	PO	16	0000 0010 1001 1100$_b$	The order is tile-resolution-layer-component-precinct
				GL	8	0000 0011$_b$	Key granularity is resolution
			V_{KT}	N_V	16 (RBAS)	3	There are 3 keys in the list
				S_V	8 (RBAS)	8	Size of each key is 8 bytes
				VL	64	*Key*0	The first key is *key*0, for resolution 0
					64	*Key*1	The second key is *key*1, for resolution 1
					64	*Key*2	The third key is *key*2, for resolution 2
		SIZ_{HMAC}			16	20	Size of MAC is 20
PD					8 (FBAS)	0$_b$	The byte aligned segment does not follow
						0$_b$	Pixel domain is not used
						0$_b$	Wavelet coefficient domain is not used
						0$_b$	Quantized wavelet coefficient domain is not used
						1$_b$	Codestream domain is used
						000$_b$	Reserved for ITU-T | ISO use
F_{PD}					8 (FBAS)	0$_b$	The FBAS byte does not follow
						0$_b$	Both packet header and body are encrypted
						0000 00$_b$	Reserved for ITU-T | ISO use
G	PO				16	0000 0101 0011 1000$_b$	The order is tile-resolution-layer-component-precinct
	GL				8	0000 0100$_b$	Tool granularity is layer
V	N_V				32 (RBAS)	9	There are 9 MACs (3 MACs per resolution)
	S_V				8 (RBAS)	20	Size of each MAC is 20 bytes
	VL				160	*M*0	The first MAC is *M*0
					160	*M*1	The second MAC is *M*1
					160	*M*2	The third MAC is *M*2
					160	*M*3	The fourth MAC is *M*3
					160	*M*4	The fifth MAC is *M*4
					160	*M*5	The sixth MAC is *M*5
					160	*M*6	The seventh MAC is *M*6
					160	*M*7	The eighth MAC is *M*7
					160	*M*8	The ninth MAC is *M*8

6.4 Distortion field examples

6.4.1 and 6.4.2 provide a few simple examples on the use of the distortion field.

6.4.1 Example 1

This example builds on the ZOI example 3 in 6.1.3 to show how distortion values can be associated with the two data segments signalled by the ZOI in that example. As discussed earlier, example 3 in 6.1.3 signalled two data segments: (1) bytes 10 to 100 and (2) bytes 10000 to 12000. To associate distortion fields to these two data segments requires two steps. First, the distortion field is signalled in DCzoi. Second, the distortion values are signalled using $Pzoi^2$. Therefore, the only changes to the ZOI example 3 in 6.1.3 are to set the distortion field bit in DCzoi, and to add $Pzoi^2$ (the final 9 lines in Table 75).

Table 75 – Associating distortion field to two data segments (extension of ZOI example 3 in 6.1.3)

Parameter			Size (bits)	Value (in order)	Derived meaning
NZzoi			8 (RBAS)	1	Number of Zones is one
$Zone^0$	DCzoi		1	0_b	The byte aligned segment does not follow
			1	1_b	Non-image related description class
			6	010001_b	Byte ranges after the SOD marker are specified and associated distortion fields are specified
	$Pzoi^2$	$Mzoi^2$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	1_b	Multiple items are specified
			2	01_b	Range mode
			2	01_b	Izoi uses 16-bit integer
			1	0_b	Izoi is described in one dimension
		$Nzoi^2$	8 (RBAS)	2	Number of data segments is 2
		$Izoi^{2,1}$	16	0000 0000 0000 1010$_b$	Starting byte location is 10th (bytes)
			16	0000 0000 0110 0100$_b$	Ending byte location is 100th (bytes)
$Zone^0$	$Pzoi^2$	$Izoi^{2,2}$	16	0010 0111 0001 0000$_b$	Starting byte location is 10000th (bytes)
			16	0010 1110 1110 0000$_b$	Ending byte location is 12000th (bytes)
	$Pzoi^6$	$Mzoi^6$	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	1_b	Multiple items are specified
			2	10_b	Index mode
			2	00_b	Izoi uses 8 bits to represent each distortion value
			1	0_b	Izoi is described in one dimension
		$Nzoi^6$	8 (RBAS)	2	Number of data segments is 2
		$Izoi^{6,1}$	8	D1 value	Distortion value for the first segment
		$Izoi^{6,2}$	8	D2 value	Distortion value for the second segment

6.4.2 Example 2

This example describes how distortion values can be associated with JPEG 2000 packets. The DCzoi specifies a range of 4 packets and the distortion field is also signalled. $Pzoi^1$ gives the range of packets and $Pzoi^2$ describes the distortion associated with each of these packets. Notice that since $Pzoi^1$ specifies a range of length 4, and $Pzoi^2$ specifies 4 values, each item in the range is associated with one value, e.g., each packet is associated with one distortion.

Table 76 – Signalling a range of packets and associating distortions for each packet

Parameter			Size (bits)	Value (in order)	Derived meaning
NZzoi			8 (RBAS)	1	Number of Zones is one
Zone[0]	DCzoi		1	0_b	The byte aligned segment does not follow
			1	1_b	Non-image related description class
			6	100001_b	Packets are specified and associated distortion fields are specified
	Pzoi[1]	Mzoi[1]	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	0_b	Single item is specified
			2	01_b	Range mode
			2	00_b	Izoi uses 8-bit integer
			1	0_b	Izoi is described in one dimension
		Nzoi[1]	8 (RBAS)	1	Number of data segments is 1
		Izoi[11]	8	$0000\ 0000_b$	Starting packet is number 0
			8	$0000\ 0011_b$	Ending packet is number 3
	Pzoi[6]	Mzoi[6]	1	0_b	The byte aligned segment does not follow
			1	0_b	The specified zones are influenced by the JPSEC tool
			1	1_b	Multiple items are specified
			2	10_b	Index mode
			2	00_b	Izoi uses 8 bits to represent each distortion value
			1	0_b	Izoi is described in one dimension
		Nzoi[6]	8 (RBAS)	4	Number of data segments is 4
		Izoi[6,1]	8	D1 value	Distortion value for the first packet
		Izoi[6,2]	8	D2 value	Distortion value for the second packet
		Izoi[6,3]	8	D3 value	Distortion value for the third packet
		Izoi[6,4]	8	D4 value	Distortion value for the fourth packet

Annex A

Guidelines and use cases

(This annex forms an integral part of this Recommendation | International Standard.)

A.1 A class of JPSEC applications

A.1.1 Introduction

This clause gives a conceptual description of how a class of JPSEC applications can be implemented. This class of application exemplifies scenarios of secure JPEG 2000 image distribution. The following clauses provide an overview of a conceptual JPSEC application including JPSEC entities and information that are communicated between them. This description is conceptual and neither intends to define a concrete implementation nor specify requirements for an implementation; specific applications might or might not include entities identified in the following description.

A.1.2 Overview of a secure JPEG 2000 image distribution

Figure A.1 shows an overview of the class of JPSEC applications of secure JPEG 2000 image distribution. In these applications, the JPSEC application can be required to provide various security services for JPEG 2000, for example, confidentiality of image exchange, authentication of image source and content.

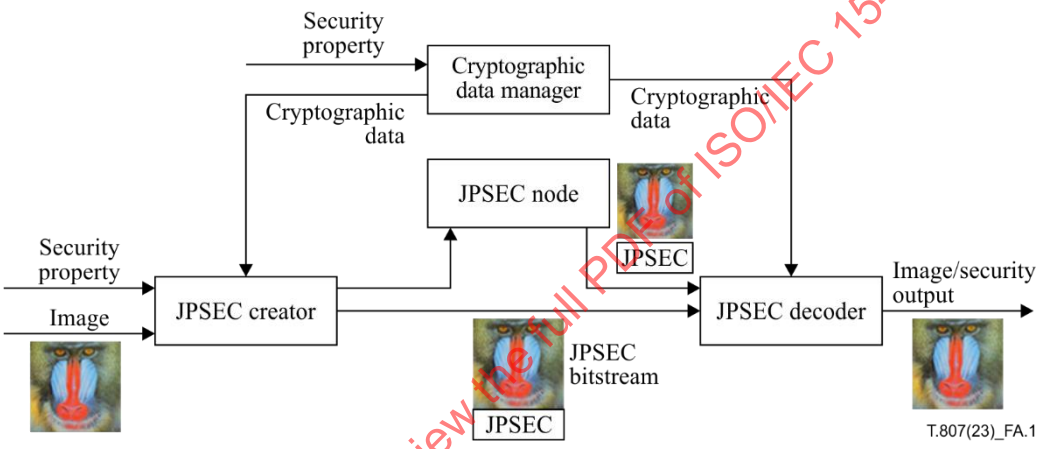


Figure A.1 – Overview of a secure JPEG 2000 image distribution application

In the secure JPEG 2000 image distribution application, one can identify the following steps:

Step 1: A JPSEC codestream is created by a JPSEC creator.

Step 2: The JPSEC codestream is distributed through some JPSEC node or nodes.

Step 3: The JPSEC codestream is received and rendered by a JPSEC consumer.

Step 1: JPSEC codestream creation

The creator is in charge of creating the secure JPEG 2000 codestream. This codestream can be created from bitmap data or from JPEG 2000 compressed data. A JPSEC creator applies various security techniques, such as encryption, signature generation, and ICV (Integrity Check Value) generation to a given image data.

To secure the image data, the creator defines which Security Parameter Property is associated to the image. A "Security Parameter Property" includes the following attributes:

– Zone of Influence (coverage area of each protection method);
– Processing Domain (domain to be processed by each protection method);
– Granularity (unit of each protection method);
– JPSEC tool identification (applied cryptographic algorithm and related parameters).

Step 2: JPSEC codestream delivery

A JPSEC codestream can be transferred to a JPSEC consumer either directly via a network or media (such as a CD-ROM). It can also be transferred through a JPSEC node which can apply various types of additional processing, such as a transcoding, to the JPSEC codestream.

When required by the JPSEC security tool methods in the Security Property Parameter of the JPSEC codestream (e.g., for encryption, or for authentication), the JPSEC creator shall distribute to the JPSEC consumer the corresponding cryptographic data through an independent ('secret') channel. This data, such as key or digital signature, can be managed either manually or automatically by a cryptographic data manager.

Step 3: JPSEC codestream consumption rendering

A JPSEC codestream is subject to a JPSEC consumer process according to the applied Security Parameter property: this implies applying the appropriate security techniques, such as decryption, authentication and integrity check. Further, for each JPSEC tool security method a JPSEC creator and JPSEC consumer can use various types of cryptographic data.

As an output of the JPSEC consumer, a decrypted image data and/or security output, such as a verification result, is produced.

The following clauses provide further detail of a conceptual JPSEC entity according to a JPSEC service. Figure A.2 shows the legend description to be used.

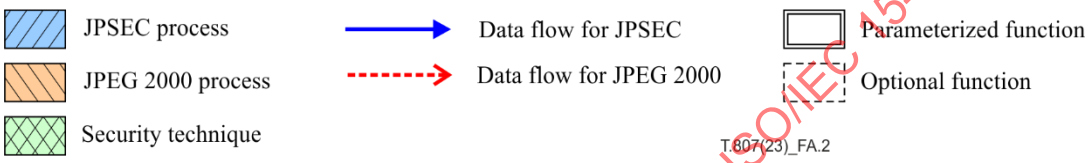


Figure A.2 – Legend description

- **JPSEC process**: A process that uses tools defined in this Recommendation | International Standard.
- **JPEG 2000 process**: A process defined in Rec. ITU-T T.800 | ISO/IEC 15444-1 (the JPEG 2000 Core coding system).
- **Security technique**: A well-known security technique either defined in this Recommendation | International Standard or in some other standard or document.
- **Data flow for JPSEC**: A data flow that communicates information defined in this Recommendation | International Standard. A dashed line indicates optional.
- **Data flow for JPEG 2000**: A data flow defined in Rec. ITU-T T.800 | ISO/IEC 15444-1 (the JPEG 2000 Core coding system).
- **Parameterized function**: A function which has several alternate functions that can be selected by an application.
- **Optional function**: A function which can be optionally applied in a JPSEC application.

A.1.3 Encryption end description procedure

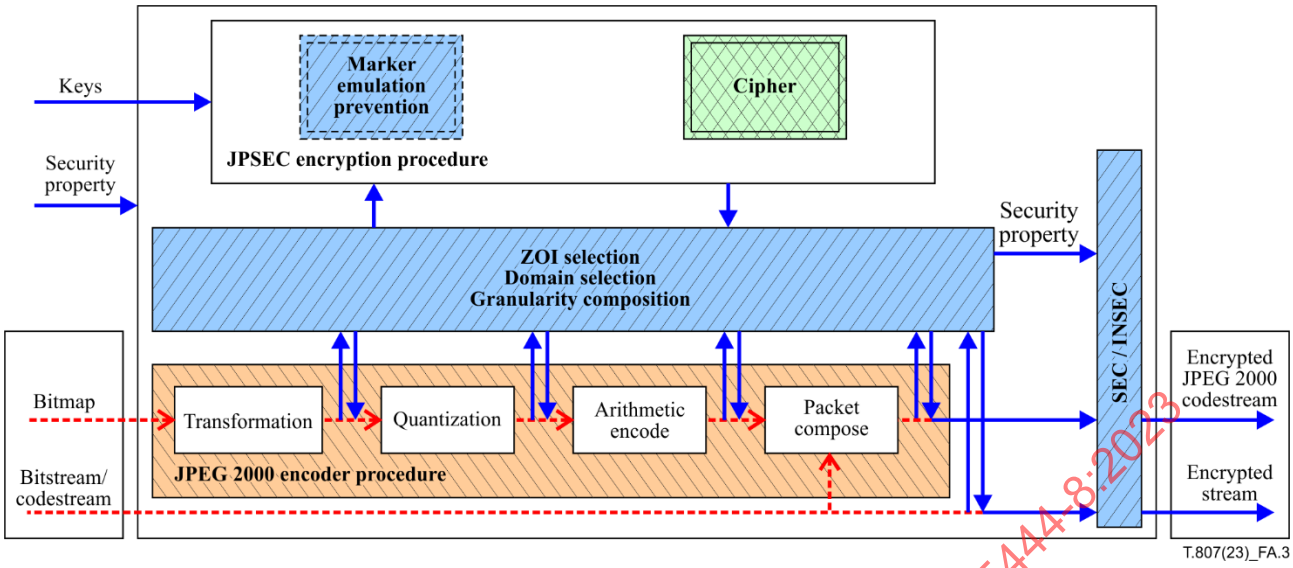


Figure A.3 – Encryption procedure

Figure A.3 shows the overview of an example encryption procedure for a JPSEC creator. This procedure includes the following processes:

- extract data according to the specified Processing Domain;
- select a portion of extracted data according to the specified Zone of Influence (i.e., a partial encryption);
- encrypt the selected data using the specified security technique. Further, it is possible to encrypt data in a unit based on the Granularity. In this case, different keys can be used for different units;
- replace the plaintext data with encrypted data;
- (optionally) apply a marker emulation prevention mechanism;
- compose the Security Parameter Property in the SEC and/or INSEC marker segment.

Note that in general the JPSEC encryption procedure generates JPSEC codestream that is not backward compatible with the JPEG 2000 Core coding system. The image data is intended to be passed on to a JPEG 2000 Core coding system compliant decoder after appropriate decryption. It is possible to apply a marker emulation prevention mechanism to avoid marker segment emulation within the encrypted codestream.

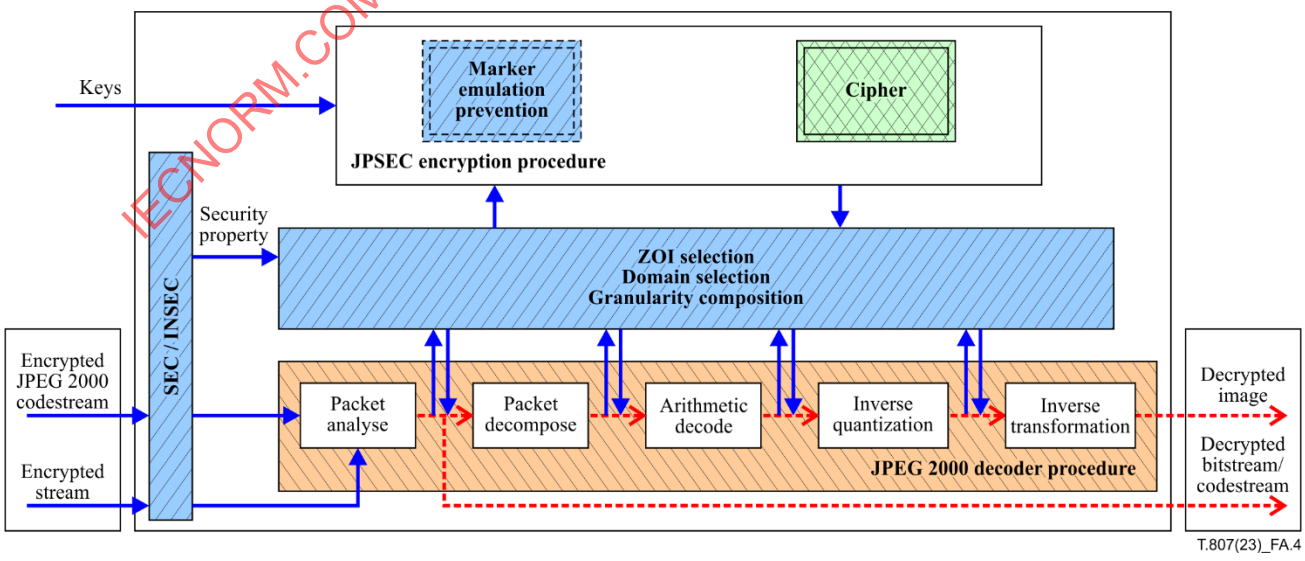


Figure A.4 – Decryption procedure

Figure A.4 shows the overview of an example decryption procedure for a JPSEC consumer. This procedure includes the following processes:

- parse the Security Parameter Property in the SEC and/or INSEC marker segment;
- extract data according to the signalled Processing Domain;
- select a portion of extracted data according to keys to be retained (i.e., a partial decryption);
- decrypt the selected data using a signalled security technique. Further, it is possible to decrypt data in a unit based on the Granularity;
- replace encrypted data with decrypted data;
- apply a marker emulation prevention mechanism, if applied at the encryption process.

A.1.4 Signature generation and authentication procedure

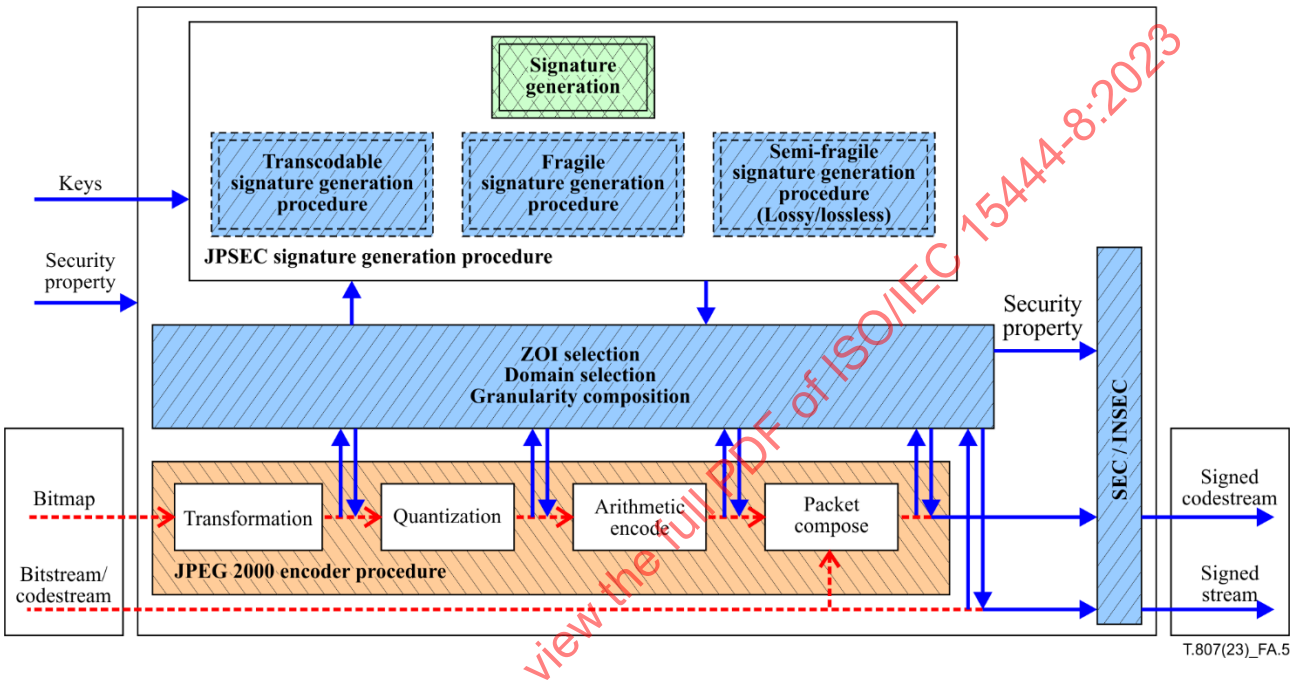


Figure A.5 – Signature generation procedure

Figure A.5 shows the overview of an example signature generation procedure for a JPSEC creator. This procedure includes the following processes:

- extract data according to the specified Processing Domain;
- select a portion of extracted data according to the specified Zone of Influence (i.e., partial signature);
- calculate digital signatures corresponding to selected data using the specified security technique. Further, it is possible to generate digital signatures in a unit based on the Granularity;
- compose the Security Parameter Property, including the calculated digital signatures, in the SEC and/or INSEC marker segment.

Note that in JPSEC, three modes of authentication are defined: "fragile mode", "semi-fragile mode (lossy/lossless)", and "transcodable mode". A "fragile mode" authentication can detect any one-bit modification for a codestream, where a "semi-fragile" mode authentication can detect any intentional detection but survive incidental distortion up to a predetermined extent. Further, a "transcoding mode" authentication can verify the source part of the codestream.

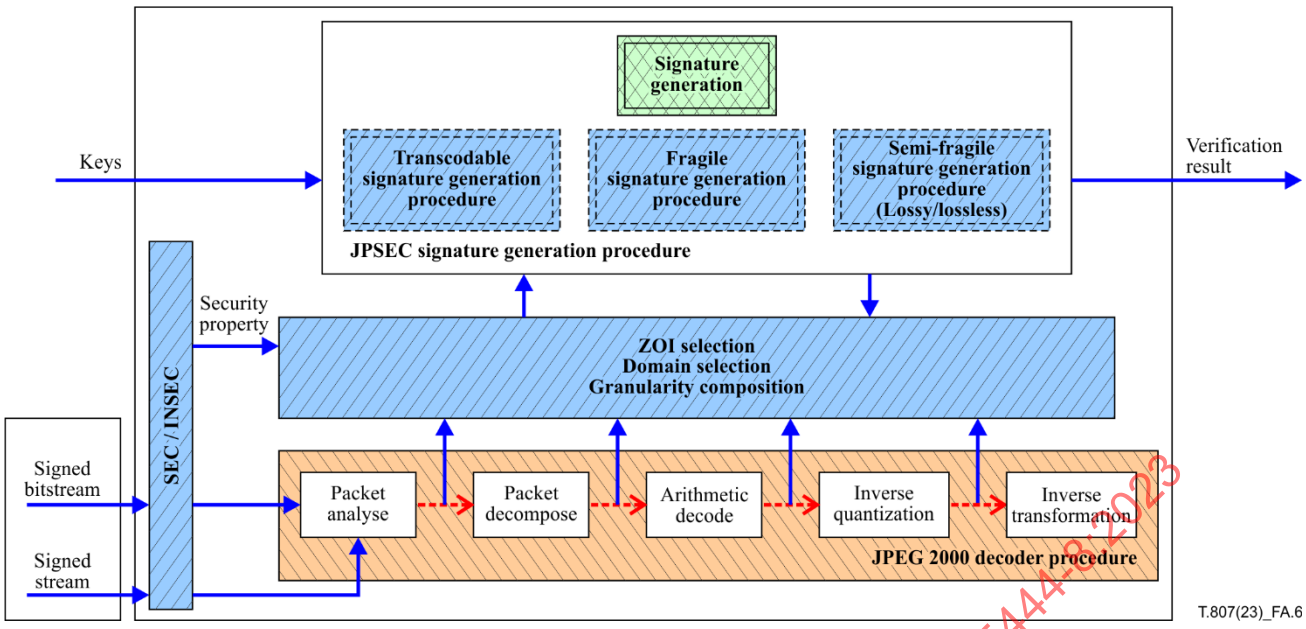


Figure A.6 – Authentication procedure

Figure A.6 shows the overview of an example authentication procedure for a JPSEC consumer. This procedure includes the following processes:

- extract data in a signalled Processing Domain;
- select a portion of extracted data according to the signalled Zone of Influence;
- verify the selected data using the signalled security technique. Further, it is possible to verify the selected data in a unit based on the Granularity.

A.1.5 Integrity check value (ICV) generation and integrity check procedure

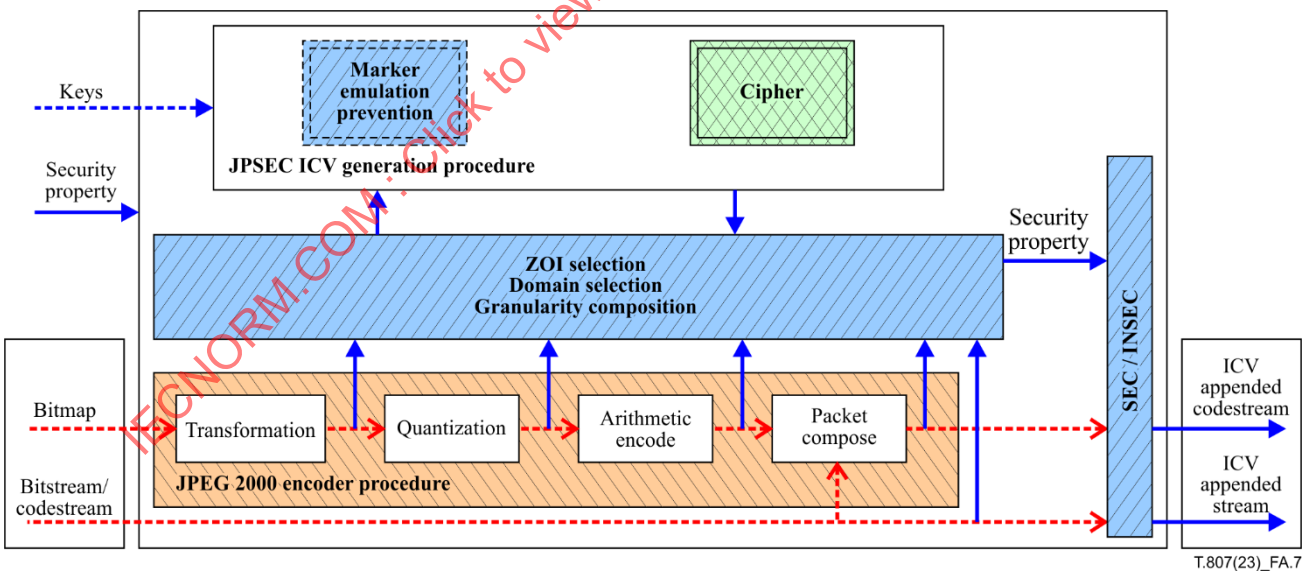


Figure A.7 – ICV generation procedure

Figure A.7 shows the overview of an example ICV generation procedure for a JPSEC creator. This procedure includes the following processes:

- extract data in a specified Processing Domain;
- select a portion of extracted data according to the specified Zone of Influence;

– calculate ICVs corresponding to selected data using the specified security technique. Further, it is possible to generate ICVs in a unit based on the Granularity;

– compose the Security Parameter Property, including the calculated ICVs, in a SEC and/or INSEC marker segment.

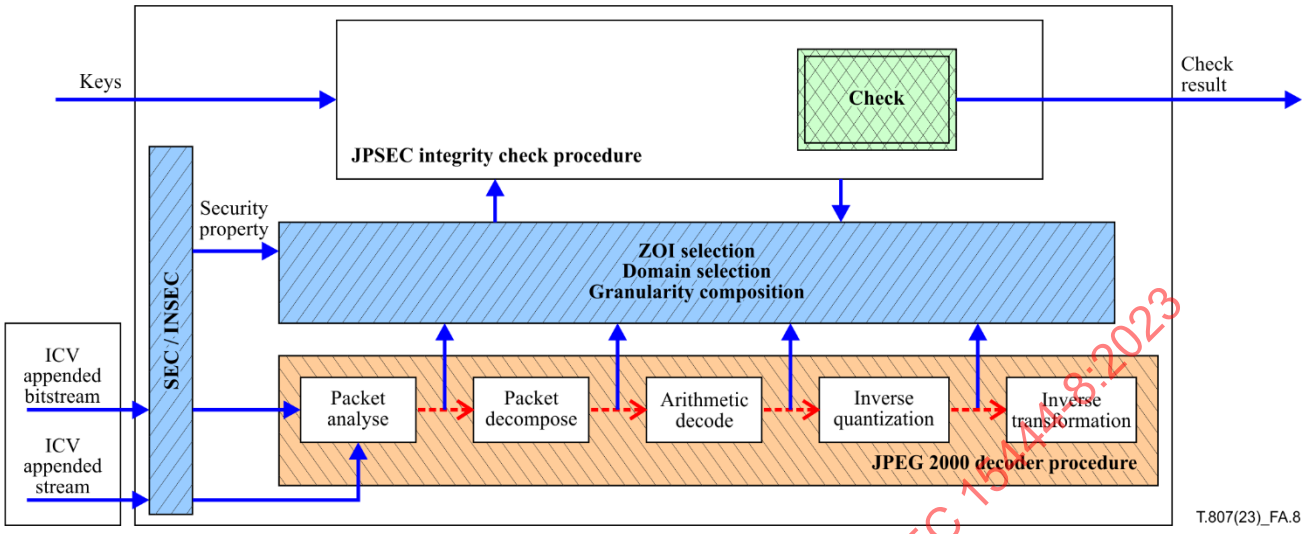


Figure A.8 – Integrity check procedure

Figure A.8 shows the overview of an example integrity check procedure for a JPSEC consumer. This procedure includes the following processes:

– extract data according to the signalled Processing Domain;

– select a portion of extracted data according to a signalled Zone of Influence;

– verify selected data using a signalled security technique. Further, it is possible to verify the selected data in a unit based on the Granularity.

Annex B

Interoperability

(This annex forms an integral part of this Recommendation | International Standard.)

B.1 Rec. ITU-T T.800 | ISO/IEC 15444-1 – Core coding system

A number of protection methods can be applied to a JPEG 2000 codestream to create JPSEC codestreams that are still strictly compliant with the JPEG 2000 Core coding system. The term "Core coding system compliance" refers to JPSEC codestreams that have a strictly defined behaviour for JPEG 2000 Core coding system decoders including those that are not aware of JPSEC.

A JPEG 2000 Core coding system decoder will skip marker segments that it does not recognize. A JPSEC tool such as the JPSEC normative tool for authentication inserts message authentication code values that are computed from the JPEG 2000 data into the SEC marker segment along with the parameters that describe the particular authentication methods that can be used by a JPSEC consumer. These parameters and values tell a JPSEC consumer how to verify that the received JPSEC codestream is authentic. Notice that the JPSEC authentication tool does not manipulate the JPEG 2000 data. Thus, a JPEG 2000 Core coding system decoder that receives this JPSEC codestream will begin decoding the JPSEC stream, it will then skip the SEC marker segment, and continue to decode the JPSEC stream as if it were a JPEG 2000 Core coding system stream. The JPSEC normative tool for authentication shares these characteristics and thus also results in a JPEG 2000 Core coding system compliant codestream.

JPSEC allows encryption and decryption to be performed on JPEG 2000 and JPSEC codestreams. When encryption is used, the JPEG 2000 data is of course changed. Strictly speaking, the Core coding system compliance is not possible with encrypted streams since it will most likely cause a JPEG 2000 Core coding system decoder to encounter disallowed values. One possible way of overcoming or at least mitigating this problem is to use the error resilience capabilities of JPEG 2000. With error resilience, it can be possible to have encrypted JPSEC codestreams that have a defined behaviour for JPEG 2000 Core coding system decoders.

JPSEC has a P_{sec} parameter field that contains security parameters for the entire codestream. This includes a flag F_{J2K} that can be set to 1 to indicate that a JPSEC codestream is decodable by JPEG 2000 Core coding system decoders. A JPSEC creator can set this parameter as it applies JPSEC tools to the JPEG 2000 codestream. It was mentioned that a JPSEC creator can accept a protected JPSEC codestream as input. If a JPSEC creator receives an input JPSEC codestream which has the F_{J2K} flag set to indicate the Core coding system compliance and then applies a JPSEC tool that loses the Core coding system compliance, it shall set the F_{J2K} flag to 0.

For JPSEC streams that are not Core coding system compliant, it is recommended to use a file extension of .jp2s to indicate that a JPEG 2000 Core coding system decoder might not be able to decode the protected codestream.

JPEG 2000 Core coding system on the extended capabilities marker segment (CAP) can be used to indicate that JPSEC is used. Specifically, the Core coding system uses R_{siz} parameter to indicate the presence of a CAP marker segment, which contains a C_{cap} parameter that can be used to signal which JPEG 2000 parts are used in the codestream. One can specify that Rec. ITU-T T.807 | ISO/IEC 15444-8 (JPSEC) is used by setting an appropriate bit in C_{cap}.

Thus, a JPSEC creator can set the R_{siz} parameter to indicate the presence of a CAP marker segment. It can insert or edit the CAP marker segment to set the C_{cap} parameter to indicate that JPSEC is used.

B.2 Rec. ITU-T T.808 | ISO/IEC 15444-9 – JPIP

B.2.1 General relationship between JPIP and JPSEC

JPIP specifies a protocol consisting of a structured series of interactions between a client and a server by means of which image file metadata, structure, and partial or whole image code streams can be exchanged in a communications efficient manner.

JPIP can be tailored via the various extensions to the JPEG 2000 file format, as defined in Rec. ITU-T T.801 | ISO/IEC 15444-2, Rec. ITU-T T.802 | ISO/IEC 15444-3 and Rec. ITU-T T.805 | ISO/IEC 15444-6. However, to achieve a simple level of interactivity that allows portions of a single JPEG 2000 file or codestream to be transferred, these other capabilities are not mandated.

Provisions have been included for the extension of the JPIP protocol to support the current JPEG 2000 Standards ITU-T Rec. T.802 | ISO/IEC 15444-3, Motion JPEG 2000, and Rec. ITU-T T.805 | ISO/IEC 15444-6, Compound Documents, and the future extensions of JPEG 2000 (currently JP3D, JPSEC, and JPWL).

JPSEC provides security services for JPEG 2000 images. The JPSEC syntax supports two types of markers: SEC and INSEC. One or more SEC markers appear in the main header of JPSEC bit stream. In other words, JPSEC consumes a

JPEG 2000 codestream, modifies the JPEG 2000 main header to form a new JPSEC "main header", and modifies the corresponding JPEG 2000 data stream to form a new protected data stream if applicable. INSEC markers can appear in the "data" portion of the data stream. It specifies some "smaller size" or "local area" parameters compared to SEC marker and can be used to complement the SEC marker.

It is observed that JPIP is just beyond the transport layer, while JPSEC is at the application layer. From this point of view, JPIP provides a transport service to JPSEC. That is, the JPIP offers efficient tools to deliver image information, including main header (all of the markers) and codestreams, between servers and clients. This clause considers how JPIP can be used to transport JPSEC content.

B.2.2 Specific issues on interactivity between JPIP and JPSEC

This clause describes the issues that a JPIP sender and receiver consider to transport JPSEC content.

In Rec. ITU-T T.808 | ISO/IEC 15444-9, both JPP- and JPT-stream media types use the main header data-bin. This data-bin consists of a concatenated list of all markers and marker segments in the main header, starting from the SOC marker. It contains no SOT, SOD or EOC markers. However, the main header of JPEG 2000 does not include a SEC marker and its segment. As a result, Rec. ITU-T T.808 | ISO/IEC 15444-9 does not specify support for the SEC marker segment specified in JPSEC. Thus, a JPIP sender and receiver shall recognize the SEC marker segment(s) that appear in the main header of a JPSEC codestream.

Rec. ITU-T T.808 | ISO/IEC 15444-9 describes its support to the precinct data. However, Rec. ITU-T T.808 | ISO/IEC 15444-9 does not specify if it supports INSEC marker and its segment specified in JPSEC. Thus, a JPIP sender and receiver shall recognize the INSEC marker segment that can appear in the data portion of a JPSEC codestream.

In Rec. ITU-T T.808 | ISO/IEC 15444-9, the tile header data-bins appear only with the JPP-stream media type. For data-bins belonging to this class, the in-class identifier holds the index (starting from 0) of the tile to which the data-bin refers. This data-bin consists of markers and marker segments for tile n. It shall not contain an SOT marker segment. Inclusion of SOD markers is optional. This data bin can be formed from a valid codestream, by concatenating all marker segments except SOT and POC in all tile-part headers for tile n.

In Rec. ITU-T T.808 | ISO/IEC 15444-9, the tile data-bins shall be used only with the JPT-stream media type. For data-bins belonging to this class, the in-class identifier is the index (starting from 0) of the tile to which the data-bin belongs. Each tile data-bin corresponds to the string of bytes formed by concatenating all tile-parts belonging to the tile, in order, complete with their SOT, SOD and all other relevant marker segments.

As mentioned above, Rec. ITU-T T.808 | ISO/IEC 15444-9 describe the support to the tile-part header and tile-part data. However, Rec. ITU-T T.808 | ISO/IEC 15444-9 do not specify if they support SEC marker segments and INSEC marker segments. Thus, a JPIP sender and receiver shall recognize and transport these marker segments along with the protected data.

B.2.3 Summary

Generally speaking, JPSEC makes itself suitable to be transported by JPIP. INSEC marker is used in the codestream to describe some "small" specific data part that is protected by security tool/tools. It makes JPSEC more flexible. To make INSEC more robust, the service layer (currently JPIP) should provide the good Quality of Service or protection on the INSEC marker and its segment. In order to achieve this goal, JPIP and JPSEC need to work out some issues and make sure the interactivity between JPIP and JPSEC.

B.3 Rec. ITU-T T.810 | ISO/IEC 15444-11 – JPWL

Wireless JPEG 2000 or JPWL (Rec. ITU-T T.810 | ISO/IEC 15444-11) extends the baseline JPEG 2000 specification to achieve the efficient transmission of JPEG 2000 imagery over an error-prone transmission environment. More specifically, JPWL defines a set of tools and methods to protect the codestream against transmission errors. It also defines means to describe the sensitivity of the codestream to transmission errors, and to describe the locations in the codestream of residual transmission errors.

JPWL is notably addressing the protection of the image header, Forward Error Correcting (FEC) codes, Unequal Error Protection (UEP), joint source-channel coding, data partitioning and interleaving, and robust arithmetic coding. JPWL is not linked to a specific network or transport protocol, but provides a generic solution for the robust transmission of JPEG 2000 imagery over error-prone networks.

The main functionalities of JPWL are:

- to protect the codestream against transmission errors;
- to describe the degree of sensitivity of different parts of the codestream to transmission errors;
- to describe the locations of residual errors in the codestream.

JPWL defines four marker segments: Error Protection Capability (EPC), Error Protection Block (EPB), Error Sensitivity Descriptor (ESD) and Residual Error Descriptor (RED).

The EPC marker segment indicates which JPWL normative and informative tools are used in the codestream. More specifically, EPC signals whether the three other normative marker segments defined by JPWL, namely the error sensitivity descriptor (ESD), the residual error descriptor (RED) and the error protection block (EPB) are present in the codestream. Furthermore, EPC signals the use of informative tools which have been previously registered with the JPWL RA. EPC is always present in a JPWL codestream.

The primary function of EPB is to protect the Main and Tile-part header. However, it can also be used to protect the remaining of the codestream. The EPB marker segment contains information about the error protection parameters and redundancy data used to protect the codestream against errors.

The ESD marker segment contains information about the sensitivity of codestream to errors. This information can be exploited when applying an Unequal Error Protection (UEP) technique. Straightforwardly, more powerful codes are used to protect the most sensitive portion of the codestream. This information can also be used for selective retransmissions. Finally, the information carried in ESD could also be used for other non-JPWL applications such as efficient rate transcoding or smart prefetching.

The RED marker segment signals the presence of residual errors in the codestream. Indeed, a JPWL decoder can fail to correct all the errors in a codestream. RED allows signalling the location of such residual errors. This information can then be exploited by a JPEG 2000 decoder in order to better cope with errors. For instance, the decoder could request retransmission, conceal the errors or discard the corrupted information.

B.3.1 General relationship between JPWL and JPSEC

The combination of JPWL and JPSEC, as shown in Figure B.1, is used whenever JPEG 2000 images need to be secured and transmitted over an error-prone wireless channel.

At the transmitter side, JPWL error sensitivity is typically generated during JPEG 2000 encoding. JPSEC tools are then applied to the codestream in order to secure it. Finally, JPWL encoding tools are used to make the codestream more robust to transmission errors.

At the receiver side, JPWL decoding tools are first applied to correct possible transmission errors. During this step, JPWL can also generate residual errors information. Finally, JPSEC tools are applied in order to fulfil the selected security services.

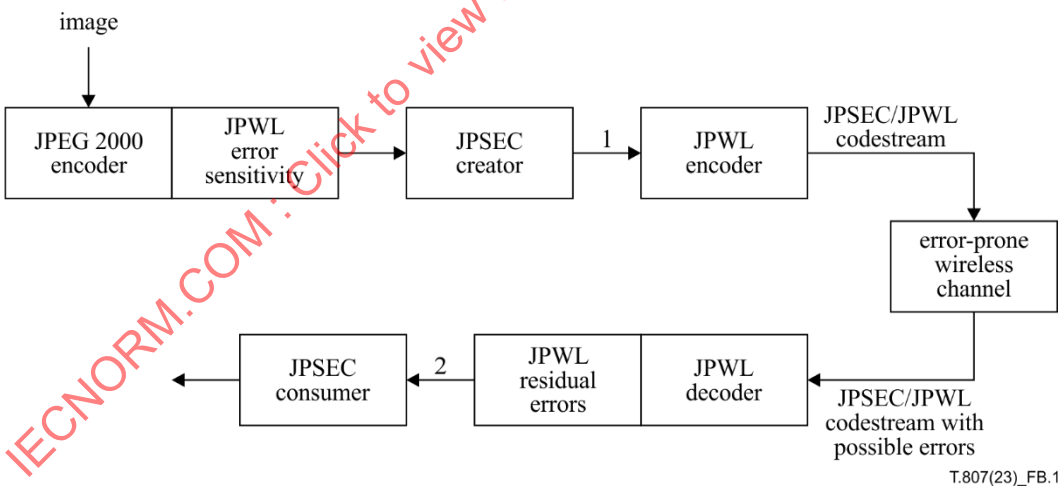


Figure B.1 – Typical JPWL and JPSEC combination

B.3.2 Specific issues on interoperability between JPWL and JPSEC

A number of issues are important to consider for interoperability between JPWL and JPSEC, as detailed hereafter:

1) JPWL Error Protection Capability (EPC): The presence of this marker segment will affect byte ranges. Note that this marker segment is always present in a JPWL codestream.
2) JPWL Error Protection Block (EPB): This marker segment is typically added as the last step at the transmitter and removed as the first step at the receiver. In principle, it should not affect JPSEC.
3) JPWL Error Sensitivity Descriptor (ESD): This marker segment is typically added during JPEG 2000 Core coding system encoding, in which case it will be transparent to subsequent JPSEC operations. However,

JPSEC could adversely affect the use of ESD in JPWL. In particular, JPSEC should not change byte ranges whenever ESD uses byte ranges. In addition, the JPSEC operations should not affect distortion values; otherwise the information carried by ESD becomes irrelevant. In the latter case, the JPSEC creator has the option to remove the ESD marker segment.

4) JPWL Residual Error Descriptor (RED): This marker segment can be inserted after JPWL decoding. It can therefore affect JPSEC byte ranges. It can also impact JPSEC authentication techniques. In case of a corrupted codestream, the RED information can be useful for a JPSEC consumer to appropriately handle it.

5) JPSEC SEC: The presence of this marker segment will affect byte ranges. Note that this marker segment is always present in a JPSEC codestream.

6) JPSEC INSEC: The presence of this marker segment will affect byte ranges. Note that this marker segment appears in the codestream data.

In the case when there are no residual errors, the JPWL encoder and decoder should ideally be transparent. In other words, in this case, the streams at points 1 and 2 in the above figure should be strictly identical.

As a general recommendation, when used in combination with JPWL, it is preferable for JPSEC to use byte ranges beginning after SOD marker in order to minimize problems with byte ranges. In addition, it is preferable to restrict the presence of JPWL marker segments to the Main header and to avoid their presence in the Tile-part headers.

Annex C

File format security

(This annex forms an integral part of this Recommendation | International Standard.)

C.1 Scope

This annex specifies JPSEC file format derived from the ISO base file format and modifications to JPEG family file format (including JP2, JPX and JPM) for protection and secure adaptation of scalable pictures, which is possibly encrypted and/or authenticated by the owner. The pictures could be either static pictures or time-sequenced pictures. In particular, this annex provides functionality to do the following:

- To store coded media data corresponding to different scalability levels. Elementary stream (ES) is used for this purpose. There are three types of ESs, self-contained ES, scalable composed ES and decodable composed ES.
- To define tracks describing the characteristics of the coded media data stored in ES. For example, the track should be able to indicate scalability level (resolution, layer, region, etc.) and the rate-distortion hints of the coded media data in order to facilitate easy and secure adaptation.
- To define new file format boxes to signal protection tools and parameters applied to coded media data or metadata. The protection tools can be applied to either static JPEG 2000 pictures or time-sequenced JPEG 2000 pictures.
- The protection tools defined in this Recommendation | International Standard can be applied to JPEG family file formats including JP2, JPX and JPM and ISO-derived file formats such as MJ2 for Motion JPEG.

C.2 Introduction

C.2.1 Security protection at file format level

This annex describes a JPSEC file format derived from the ISO base file format and modifications to JPEG family file format, to add security protection to JPEG 2000 pictures at the file format level. The protection applied at the file format level can be classified into two types: item-based protection and sample-based protection, both structures are defined by the ISO base file format. The item-based protection is designed to protect any byte ranges (including coded media data and metadata) while the sample-based protection is designed to protect time-sequenced media including JPEG 2000 pictures.

When the security tools applied change the data length, it shall update all pointers and length fields in all boxes, to ensure correct parsing by the reader.

C.2.2 Item-based protection

This annex describes two item-based protection schemes in the ISO base file format, by leveraging the syntax and structures specified by this Recommendation | International Standard. Specifically, it describes schemes for decryption and authentication. Each item in the ItemLocationBox is protected by one or more protection schemes in the ItemProtectionBox. When multiple schemes are used (or chained together), the order in which they are applied can be significant and thus shall be specified. This annex also specifies how such operations should be chained together. In addition, the ItemDescriptionBox and ItemCorrespondingBox are added into the ISO base file format to allow the flexible processing properties that are provided by JPSEC. Specifically, the ItemDescriptionBox allows media-dependent metadata (such as resolution, quality layer, spatial region, and color space component) to be associated with different portions of the file. These descriptions can be provided regardless of whether protection is applied. When used with scalable coded pictures, this allows the file to be scaled down or transcoded without parsing or decoding the media data. In cases where protection is applied, this provides the benefit of enabling transcoding without requiring decryption.

C.2.3 Sample-based protection of scalable media

For time-sequenced pictures, this annex adds syntaxes to facilitate scalability at the file format level, including scalable composed elementary stream (ES), decodable composed ES, pointer structure, container structure and ByteData structure. The scalable coded pictures can be divided (either physically or virtually) into elementary streams at different scalability level, such that the adaptor/transcoder can "thin" media data with low complexity.

Figure C.1 gives an overview of the file format specified by this annex and also shows how the specified FF is used to adapt the media data.

Given a sequence of JPEG 2000 pictures (also referred to as *Self-contained ES*), there are two approaches to construct the file format. In the first approach, the MDAT box contains one or more *Scalable Composed ESs*, each of which corresponds to one scalability level of the media data, e.g., a resolution or a layer. The scalable composed ES shall be stored in MDAT box that is co-located with the file format. The self-contained ES can be located in either MDAT box in the same file, or a different file whose format is not specified in this Recommendation | International Standard. The scalable composed ES might not be decodable by itself, it might need to be combined with other scalable composed ESs to generate fully decodable JPEG 2000 pictures. In the second approach, the MDAT box contains one or more *Decodable Composed ESs*, and each ES constitutes fully decodable JPEG 2000 pictures by itself. Similarly, decodable composed ESs shall be stored in MDAT box co-located with the file format, and the self-contained ES can be stored in either MDAT box in the same file, or a different file whose format is not specified by this annex.

Each scalable composed ES or decodable composed ES shall be described by at least one track. The characteristics of the ES (like resolution, layer, and region) are indicated in SampleEntryBox inside each track.

To generate a fully decodable JPEG 2000 codestream from scalable composed ESs, a JPEG 2000-aware adaptor should have the capability to dynamically generate the image headers (based on the number of resolutions, layers and region in the adapted codestream), to insert empty packets or to insert POC markers as needed to make the resulting codestream decodable by any standard decoder. However, if a simple adaptor is used, the resulting codestream can have an inconsistent image header and there can be a missing packet, which require a JPEG 2000 adaptive-format decoder.

As a decodable composed ES is decodable by itself, a simple adaptor is sufficient to generate fully compliant JPEG 2000 pictures.

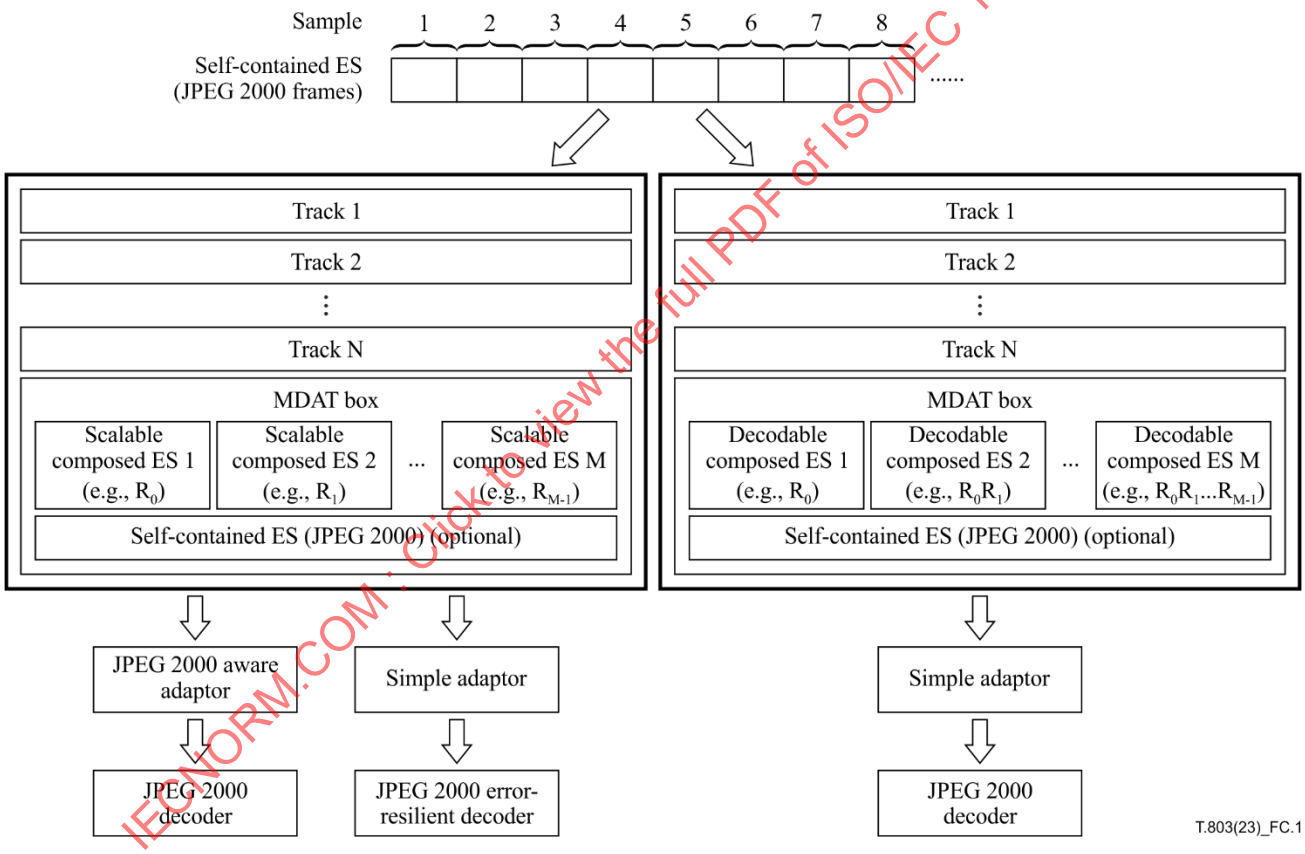


Figure C.1 – System diagram for time-sequenced scalable media

Each elementary stream is described by at least one media track, and its characteristics are described in SampleDescriptionEntry or SampleGroupEntryBox within the track. It is possible that a single elementary stream is described by multiple tracks, each of which can describe different aspects of the elementary stream.

The sample-based protection can be applied to all samples or a group of samples in a scalable composed ES or decodable composed ES. If protection is applied to all samples, a ProtectionSchemeInfoBox signalling the parameters of the protection tool is added to the SampleDescriptionBox, which is then encapsulated as described in C.5.2. In addition, if protection is applied to a group of samples, a ProtectionSchemeInfoBox is added to their SampleGroupEntryBox, which is then encapsulated as described in C.5.4.

C.3 Extension to ISO base media file format

C.3.1 Overview

C.3.2 to C.3.10 document technical extensions (additional box types) to the ISO based media file format, which could be used for protection, adaptation, or secure adaptation of scalable coded pictures. However, the added box types could be used for other purposes as well. In particular, this clause defines ProtectionSchemeInfoBox for the decryption tool and authentication tool, ItemDescriptionBox, ScalableSampleDescriptionEntry, ScalableSampleGroupEntry, and Generic Protected Box. All other boxes defined in ISO/IEC 14496-12 are still used as is.

C.3.2 Incorporate JPSEC codestream into ISO-driven file format

A JPSEC codestream can be placed as a payload in the 'mdat' box of the ISO base file format. In the Sample Description Box ('stsd'), the 'codingname' of the corresponding Sample Entry is defined to be 'jpsc', which is a registered identifier for JPSEC decoder. In this case, the security service is provided by JPSEC at codestream.

C.3.3 Protected file format brand

Files conforming to this Recommendation | International Standard can use 'ffsc' as the major brand in the File Type Compatibility Box.

Files conforming to this Recommendation | International Standard, i.e., containing protection or authentication information can use 'ffsc' as a compatible brand in the File Type Compatibility Box.

There are uses of this Recommendation | International Standard which are compatible with JP2, JPX, MJ2, and JPM files. A typical use of this Recommendation | International Standard will leave the major brand of a file unchanged, but add boxes and thus add 'ffsc' as a compatible brand.

Thus, brands including *'isom'*, *'iso2'*, *'jp2\040*, *'jpx\040'* and *'jpm\040'* should be compatible.

The *'ffsc'* compatible brand indicates the use of new boxes and new tools corresponding to the protection methods in JPSEC.

A file that has been protected, to the extent that an application intending to process the JP2, JPX, JPM, or other file type content will be unable to do so without using protection tools, can use the 'ffsc' major brand as the file type; such a protected file shall not use a major brand for which it is no longer conformant.

C.3.4 Summary of boxes used

The ISO base media file format defines two structures to describe a presentation: the logical structure and media sequence structure. The logical structure uses the ItemLocationBox ('iloc') to describe an item which is the byte range or a series of byte ranges for a particular file, either a local file or a remote file. The media sequence structure uses the SampleGroupDescriptionBox ('spgd') or SampleDescriptionBox ('stsd') to describe the samples, which could be a frame of video, a time-contiguous series of video frames, or a time-contiguous compressed section audio.

Accordingly, the protection in the ISO base media file format level is classified into item-based protection and sample-based protection, as described in C.5.2 and C.5.4, respectively.

Several boxes are used from ISO/IEC 14496-12, these are marked as "Existing" in Table C.1. Boxes defined in this Recommendation | International Standard are listed as "New" in Table C.1. The definitions for these boxes depend on the definitions of Box and FullBox from ISO/IEC 14496-12, which are repeated for convenience in C.7.

Table C.1 – List of existing and new boxes

Box names						Status	Remarks
meta						Existing	Metadata
	iloc					Existing	Item location
	ipro					Existing	Item protection
		sinf				Existing	Protection scheme information box
			frma			Existing	Original format box
			schm			Existing	Scheme type box
			schi			Existing	Scheme information box
				gran		New	Granularity box
				vall		New	Value List box
				bcip		New	Block cipher box

Table C.1 – List of existing and new boxes

Box names						Status	Remarks
					keyt	New	Key template box
				scip		New	Stream cipher box
					keyt	New	Key template box
				auth		New	Authentication box
					keyt	New	Key template box
	iinf					Existing	Item information box
	ides					New	Item description box
		dest				New	Description type box
		desd				New	Description data box
		vide				New	Visual item description entry
		j2ke				New	JPEG 2000 item description entry
	icor					New	Item correspondence box
…	…	…	…	…	…	…	…
stbl						Existing	Sample table box
	stsd					Existing	Sample description box
		ScalableSampleDescriptionEntry				New	Scalable sample description entry
	sbgp					Existing	Sample to group box
	sgpd					Existing	Sample group box
		ScalableSampleGroupEntry				New	Scalable sample group entry
gprt						New	Generic Protected box

C.3.5 Decryption scheme

The Decryption protection scheme is identified in SchemeTypeBox as follows:

```
scheme_type="decr"

scheme_version=0

scheme_uri=null
```

For the Decryption protection scheme, the structure of SchemeInfoBox is as follows:

```
aligned(8) class GranularityBox() extends Box('gran') {
     unsigned int(8) granularity;
}
```

Semantics:

`granularity` is used for item-based protection. For item-based protection, 0 indicates that the processing unit is the entire item and 1 indicates that the processing unit is one extent within an item. For sample-based protection, 0 indicates that the processing unit is all samples in the track or sample group and 1 indicates that the processing unit is one sample.

```
aligned(8) class ValueListBox() extends Box('vall') {
     unsigned int(8) value_size;
     unsigned int(16)     value_count;
     unsigned int(16)     count [value_count];
     unsigned char(value_size) value[value_count];
}
```

Semantics:

`value_size` is the size in bytes of each value in the array.

`value_count` is the number of (`count, value`) pairs in the array. For item-based protection, the (`count, value`) pairs are used to map each value to `count` processing units. For sample-based protection, the (`count, value`) pairs are used to map each value to `count` samples. For instance, the `value[0]` corresponds to the first `count[0]` sample or units, and the `value[1]` corresponds to the next `count[1]` samples or units, and so on.

```
aligned(8) class KeyTemplateBox() extends Box('keyt') {
      unsigned int(16)      key_size;
      unsigned int(8) key_info;
      GranularityBox()      GL;             //optional
      ValueListBox()  VL;
}
```

Semantics:

`key_size` is the size of key in bits.

`key_info` indicates the meaning of the values in the ValueListBox. 1 means the values are ITU-T X.509 certificate; 2 means the values are URIs for certificate or secret keys.

`GL` is a **`GranularityBox`**`()`.

`VL` is a **`ValueListBox`**`()`, containing a list of values, whose meaning is defined by the `key_info` field.

```
aligned(8) class BlockCipherBox() extends Box('bcip') {
      unsigned int(16)      cipher_id;
      bit(6)          cipher_mode;
      bit(2)          padding_mode;
      unsigned int(8) block_size;
      KeyTemplateBox()      KT;
}
```

Semantics:

`cipher_id` identifies which block cipher algorithm is used for protection. Values are defined in Table 25.

`cipher_mode` could be ECB, CBC, CFB, OFB or CTR. Values are defined in Table 29.

`padding_mode` is ciphertext stealing or PKCS#7-padding. Values are defined in Table 30.

`block_size` is size of block for block cipher.

`KT` is a **`KeyTemplateBox`**`()`, holding all the key information used by the block cipher.

```
aligned(8) class StreamCipherBox() extends Box('scip') {
      unsigned int(8) cipher_type;
      unsigned int(16)      cipher_id;
      KeyTemplateBox()      KT;
}
```

Semantics:

`cipher_type` indicates the type of cipher used. It has values `cipher_type` = STRE for stream cipher or `cipher_type` = ASYM for asymmetric cipher.

`cipher_id` identifies the stream cipher algorithm used for the protection. If `cipher_type` = STRE, see Table 26; if `cipher_type` = ASYM, see Table 27.

`KT` is a **`KeyTemplateBox`**`()`, holding all the key information used by the stream cipher.

```
aligned(8) class SchemeInfomationBox() extends Box('schi', cipher_id) {
      unsigned int(8) MetaOrMedia;
      unsigned int(8) HeaderProtected;
      BlockCipherBox(); or  StreamCipherBox();
      GranularityBox()      GL;
      ValueListBox()  VL;
}
```

Semantics:

`MetaOrMedia` is to indicate whether the protected data segment corresponds to media data segment or meta data boxes. (0 for media data and 1 for meta data boxes).

`HeaderProtected` is to indicate whether the protection is applied to the box content only (value 0) or applied to the whole box including its header (value 1).

The SchemeInformationBox can contain a **`BlockCipherBox`**`()`, or a **`StreamCipherBox`**`()`, which are containers for the parameters of the cipher algorithms. These boxes can only contain particular `cipher_id` values.

GL is a **GranularityBox**(), holding information about the processing unit. This field is optional for sample-based protection and required for item-based protection.

VL is a **ValueListBox**(). For block cipher, this box can be empty. For stream cipher, this box contains all the initial vectors.

C.3.6 Authentication scheme

The Authentication protection scheme is identified by SchemeTypeBox as follows:

```
scheme_type="auth"

scheme_version=0

scheme_uri=null
```

For Authentication protection scheme, the structure of SchemeInfoBox is defined as follows:

```
aligned(8) class AuthBox() extends Box('Auth') {
      unsigned int(8) auth_type; //hash, cipher, or signature
      unsigned int(8) method_id;
      unsigned int(8) hash_id;
      unsigned int(16)      MAC_size;
      KeyTemplateBox()      KT;
}
```

Semantics:

auth_type indicates authentication type, including hash-based (HASH), cipher-based (CIPH) and signature-based (SIGN) authentication.

method_id identifies the authentication method. 1 indicates HMAC. If auth_type = HASH, see Table 36; if auth_type = CIPH, see Table 39; if auth_type = SIGN, see Table 41.

hash_id identifies the hash function used. If auth_type = HASH or SIGN, see Table 37; if auth_type = CIPH, see Table 25.

MAC_size is size of MAC (if auth_type = HASH or CIPH) or digital signature (if auth_type = SIGN) in bits.

KT is a **KeyTemplateBox**(), holding all the key information for the authentication.

```
aligned(8) class SchemeInfoBox() extends Box('schi', auth_method) {
      unsigned char (8)     auth_method;
      AuthBox()       authBox;
      GranularityBox()      GL;             //optional
      ValueListBox()  VL;
}
```

Semantics:

auth_method identifies the authentication method used. 0 is for hash-based MAC; 1 is for cipher-based MAC; 2 is for digital signature. Depending on the auth_method, this box could contain either **HashAuthBox**(), **CipherAuthBox**(), or **SignatureAuthBox**().

GL is a **GranularityBox**(). The field is optional for sample-based protection and required for item-based protection.

VL is a **ValueListBox**(), holding all the MACs or signatures.

C.3.7 ItemDescriptionBox

In the Item Location Box, all the items are specified as byte ranges (using the offset and length). The 'iloc' box does not contain the content-related information about the specified byte ranges, which is used by some protection methods. For instance, if secure transcoding method wants to discard the least important layer or resolution, it has to know which byte ranges correspond to the to-be-discarded layer or resolution.

As such, this clause defines two new boxes to enable the content-related processing at the file format level: Item Description Box ('ides') and Item Correspondence Box ('icor'). The 'ides' box specifies the content-related information, like layer, resolution (for visual content), period (audio content), and so on. The 'icor' box links the content-related information to the items in 'iloc' box, in the same way as the 'iinf' box links 'iloc' box to 'ipro' box.

The Item Description box ('ides') is defined as follows:

```
aligned(8) class ItemDescriptionBox() extends Box('ides') {
      unsigned int(32)      entry_count;
      for(i=0; i < entry_count; i++) {
            DescriptionTypeBox()       desType;
            unsigned int(32)           item_ID;
            DescriptionDataBox()       desData;
      }
}
```

Semantics:

`entry_count` is the number of entries in the **`ItemDescriptionBox`**().

`item_ID` references an Item in **`ItemLocationBox`**(). If this field is 0, then `item_ID` will be specified by **`ItemCorrespondenceBox`**().

```
aligned(8) class DescriptionTypeBox() extends Box('dest') {
      Unsigned int(32)      description_type;
      Unsigned int(32)      description_version;
      Unsigned int(8) description_uri[];
}
```

Semantics:

`description_type` is the 4CC code defining the description scheme.

`description_version` is the version of the description.

`description_uri` allows for the options of directing the user to a webpage if they do not have the description definition installed on their system. It is an absolute URI formed as a null-terminated string in UTF-8 characters.

```
Aligned(8) class DescriptionDataBox() extends Box('desd') {
      Box() description_specific_data [];
}
```

Semantics:

If `description_type` = 'vide', this is a **`VisualItemDescriptionEntry`**(); if `description_type` = `'j2ke'`, this is a **`J2KItemDescriptionEntry`**(); for any other value of `description_type`, the syntax of the description can be found at `description_uri`.

The **`VisualItemDescriptionEntry`**() is defined as follows:

```
aligned(8) Class VisualItemDescriptionEntry extends Box('vide') {
      unsigned int(16)      layer_start;
      unsigned int(16)      layer_count;
      unsigned int(16)      res_start;
      unsigned int(16)      res_count;
      unsigned int(16)      horizontal_offset;
      unsigned int(16)      horizontal_length;
      unsigned int(16)      vertical_offset;
      unsigned int(16)      vertical_length;
      unsigned int(16)      color_space;
      unsigned int(16)      time_start;
      unsigned int(16)      time_length;
}
```

Semantics:

The `layer_start` and `layer_count` together specify the range of layers. When `layer_start` equals to 2^{16}–1, the layer range will start from layer 0; when `layer_count` equals to 2^{16}–1, the layer range will end at the last layer. When both `layer_start` and `layer_count` equal to 2^{16}–1, the layer range will include all layers.

The res_start and res_count together specify the range of resolutions. The semantics is the same as layer_start and layer_count when their value equals to 2^{16}–1.

The horizontal_offset, horizontal_length, vertical_offset and vertical_length together specify the spatial area. The semantics is the same as layer_start and layer_count when their value equals to 2^{16}–1.

The color_space specifies the color space. 0: red color space; 1: green color space; 2: blue color space.

The intersection is applied to the layer ranges, resolution ranges, areas and color space to get the portion of the image/video specified by the **VisualItemDescriptionEntry**().

The **J2KItemDescriptionEntry**() is defined as follows:

```
aligned(8) class J2KItemDescriptionEntry() extends Box('j2ke') {
     VisualItemDescriptionEntry()    visualDesEntry; //optional
     unsigned int(16)      tile_start;
     unsigned int(16)      tile_count;
     unsigned int(16)      precinct_start;
     unsigned int(16)      precinct_count;
     unsigned int(16)      j2k_packet_start;
     unsigned int(16)      j2k_packet_count;
}
```

Semantics:

visualDesEntry() specifies image/video-specific attributes. It is optional.

tile_start and tile_count specify the tiles.

precinct_start and precinct_count specify the precincts.

j2k_packet_start and j2k_packet_count specify the JPEG 2000 defined packets.

Similar to **VisualItemDescriptionEntry**(), the intersection is applied to tiles, precincts and JPEG 2000 packets to get the portion of the JPEG 2000 codestream specified by **J2KItemDescriptionEntry**().

The **ItemCorrespondenceBox**('icor') is defined as follows:

```
aligned(8) class ItemCorrespondenceEntry() extends Box('icor') {
     unsigned int(16)      item_ID;
     unsigned int(16)      desc_ID;
}
```

Semantics:

item_ID is pointing to one item in the **ItemLocationBox**().

desc_ID is pointing to one description entry in the **ItemDescriptionBox**().

C.3.8 ScalableSampleDescriptionEntry Box

The **ScalableSampleDescriptionEntry**() is used to describe characteristics associated with scalable composed ES or decodable composed ES, like resolution levels, quality layers, cropped region. For scalable composed ES, res and layer indicate the media data of that particular resolution and layer, while for decodable composed ES, they indicate the highest resolution and highest quality layer.

When the media samples are protected by a protection tool, the **ScalableSampleDescriptionEntry**() is encapsulated as follows:

The four-character-code of the **ScalableSampleDescriptionEntry**() is replaced with another four-character-code indicating protection encapsulation, which varies only by media type, as defined below:

- Encv: to indicate that video samples are encrypted and thereby un-protection shall be applied to get meaningful media data.
- Autv: to indicate that video samples are authenticated, and the media data can still be meaningful before un-protection.
- Enct: to indicate that text samples are encrypted.
- Autt: to indicate that text samples are authenticated.
- Encs: to indicate that system samples are encrypted.
- Auts: to indicate that system samples are authenticated.

A **ProtectionSchemeInfoBox**() is added to the **ScalableSampleDescriptionEntry**(), leaving all other boxes unmodified.

The original sample entry type is stored in the **OriginalFormatBox**() within the **ProtectionSchemeInfoBox**().

```
class ScalableSampleDescriptionEntry(codingname) extends VisualSampleEntry(codingname) {
      Unsigned int(8) res;
      Unsigned int(8) layer;
      Unsigned int(32)      cropped_width, cropped_height;
      If(cropped_width > 0 && cropped_height > 0) {
            Unsigned int(32)      startx;
            Unsigned int(32)      starty;
      }
      ProtectionSchemeInfoBox() protectionSchemes[]; //optional
}
```

Semantics:

codingname is "sces" if the track is referring to a scalable composed ES and "dces" if the track is referring to a decodable composed ES.

res is the resolution of the described samples. A value of –1 indicates all resolution levels.

layer is the quality layer of the described samples. A value of –1 indicates all quality layers.

cropped_width is the width of the cropped region.

cropped_height is the height of the cropped region.

startx & starty indicate the position of the top-left corner of the cropped region. If either cropped_width or cropped_height is zero, the startx and starty will not be present.

protectionSchemes is a list of **ProtectionSchemeInfoBoxes**() to indicate the protection tools applied to the described samples. The un-protection process has to follow the order in which it appears in the list.

C.3.9 ScalableSampleGroupEntry Box

A track can be made up of samples with different characteristics and protected by different protection tools or with different parameters. The ScalableSampleGroupEntry box is used to signal the grouping of samples based. For example, if a track has 1000 samples where the first 500 samples are encrypted with Key 1 and the second 500 samples are encrypted with Key 2, the **SampleGroupDescription**() Box contains two **ScalableSampleGroupEntry**() boxes: the first box describes the first 500 samples while the second box describes the second 500 samples.

When the media samples are protected by a protection tool, the **ScalableSampleGroupEntry**() is encapsulated in the same way as **ScalableSampleDescriptionEntry**() in C.3.8.

```
class ScalableSampleGroupEntry(type) extends VisualSampleGroupEntry(type) {
      unsigned int(8) res;
      unsigned int(8) layer;
      unsigned int(32)      cropped_width, cropped_height;
      If(cropped_width > 0 && cropped_height > 0) {
            unsigned int(32)      startx;
            unsigned int(32)      starty;
      }
      ProtectionSchemeInfoBox() protectionSchemes[]; //optional
}
```

Semantics:

type indicates the grouping type. If the grouping is based on different protection tools applied to the samples, the type is "prot"; if the grouping is based on different media characteristics (like resolution or layers), the type is "attr".

res is the resolution of the described samples. A value of –1 indicates all resolution levels.

layer is the quality layer of the described samples. A value of –1 indicates all quality layers.

cropped_width is the width of the cropped region.

cropped_height is the height of the cropped region.

`startx & starty` indicate the position of the top-left corner of the cropped region. If either `cropped_width` or `cropped_height` is zero, the `startx` and `starty` will not be present.

`protectionSchemes` is a list of **`ProtectionSchemeInfoBox`**`()`s to indicate the protection tools applied to the described samples. The un-protection process has to follow the order in which it appears in the list.

C.3.10 Generic Protected Box

C.3.10.1 Definition

Box Types: 'gprt'

Container: File or any container box

Mandatory: Yes, this shall be present when a protection scheme is applied to a box that would prevent parsing the file

Quantity: Any number

The **`JProtectedBox`**`()` is used when a protection scheme is applied to a box, and use of the protection scheme prevents parsing of the box. For example, if the contents of a superbox are encrypted, including the box lengths and types, that box is no longer parsable, and thus fails to meet the original definition for the box. In this case, a **`JProtectedBox`**`()` can be placed in the file in the place of the box that no longer parses correctly, and the encrypted data placed in the **`JProtectedBox`**`()`.

Interpreting the content of the data[] portion of this box shall be done using the **`ItemLocationBox`**`()`.

Once all of the protection methods have been operated on from the **`ItemInformationBox`**`()`, the contents can be reorganized into original boxes. The first `size[0]` bytes of the unprotected `data[]` array are placed in a box of `type[0]`, and the next `size[i]` bytes are placed in a box of `type[i]`, and so on. Note that when `size_flag` is 0, the size of the original box is not disclosed and when `type_flag` is 0, the type of the original box is not disclosed. This is to prevent known-plaintext attacks. When both the `size_flag` and `type_flag` are 0, `total_size` provides the total size of the protected content.

If the entry count is 0, or if there is data left over, then that data shall be in the format of valid boxes with type and size codes, i.e., the unprotected data contains the types and sizes.

If any JPEG 2000 Core coding system mandatory box is encrypted, the "jp2" brand should be removed from the compatible list in file type box.

C.3.10.2 Syntax

```
aligned(8) class JProtectedBox() extends Box('gprt') {
      bit(1)      type_flag;
      bit(1)      size_flag;
      bit(1)      location_flag;
      unsigned int(5) reserved; // for ITU-T | ISO use
      if(size_flag == 1 || type_flag == 1 || location_flag == 1) {
            unsigned int(32)      entry_count;
            if(location_flag == 1)
                  unsigned int(8) offset_size;
            for(i=0; i<entry_count; i++) {
                  if(size_flag == 1)
                        unsigned int(32)      size;
                  if(type_flag == 1)
                        unsigned int(32)      type = boxtype;
                  if(size_flag == 1 && size == 1)
                        unsigned int(64)      large_size;
                  if(location_flag == 1)
                        unsigned int(offset_size*8)      offset;
            }
      }
      else {
            Unsigned int(32)      total_size;
            if(total_size == 1)
                  unsigned int(64)      large_total_size;
      }
      unsigned int(8) data[];
}
```

C.3.10.3 Semantics

`size_flag` indicates whether or not size value is present. A value of 1 means the size of each entry in Generic Protected Box is present; 0 means the size is not present.

`type_flag` indicates whether or not the box type is present. A value of 1 means the original type of each entry in Generic Protected Box is present; 0 means the type is not present.

`location_flag` indicates whether or not the location is present. A value of 1 means the original location of each entry in Generic Protected Box is present; 0 means the original location is not present.

`entry_count` is the number of entries in the Generic Protected Box.

`offset_size` is the length of the `offset` in bytes.

Each `size` entry is the size of a box that has been replaced by the Generic Protected Box.

Each `type` entry is the original type of a box that has been replaced by the Generic Protected Box.

Each `offset` entry is the offset of a box that has been replaced by the Generic Protected Box.

`total_size` is the total size of the entries in the Generic Protected Box.

`data[]` is an array of bytes to the end of the box that can be referenced by the ItemLocationBox and thus protected with a method defined in the ItemProtectionBox.

C.4 Elementary stream and sample definitions

C.4.1 Overview

This clause defines the structure of an elementary stream (ES).

An elementary stream contains media data, ByteData structure, container structure, pointer structure, or any mixture of the above. There are two types of ES: One type is called *self-contained ES*, which does not contain any container, pointer or ByteData structure. The format of self-contained ES is not defined in this Recommendation | International Standard, and it can be any media format like Motion JPEG 2000 sequence. The other type is referred to as *composed ES*, where the media data is composed from other elementary streams. A composed ES can either copy data from other tracks or reference data in other tracks. When a media data segment is copied from other tracks, the segment shall be wrapped in ByteData structure; the pointer structure is used to reference media data segments in other ESs. When a sample in composed ES consists of more than one structure, a container structure shall be used to wrap all structures belonging to that sample.

The composed ES can be classified into *Scalable Composed ES* and *Decodable Composed ES*. The former is designed to support scalability, i.e., to make the codestream "thinable", the adaptor can make a codestream by combining multiple such ESs. However, it is assumed that the adaptor should be able to generate new headers and modify the index number within each packet. The decodable composed ES is designed for simple adaptor that has no capability to generate headers or modify packet index numbers. The adaptor simply follows the instructions in a decodable composed ES to generate a complete decodable codestream.

Figure C.2 illustrates the relationship between self-contained ES and scalable composed ES. In scalable composed ES, each sample is wrapped by a container structure, where a pointer is referring to the desired data segment(s) in the self-contained ES. Note that a scalable composed ES does not contain any headers, the adapter has to dynamically generate the header to make a codestream that is decodable by normal decoder. In addition, each scalable composed ES might not constitute a complete decodable codestream by itself, and multiple scalable composed ESs can be required to make a complete codestream. For example, in Figure C.2, the adaptor has to combine the scalable composed ES 0 and 1 to make a resolution-1 JPEG 2000 frame.

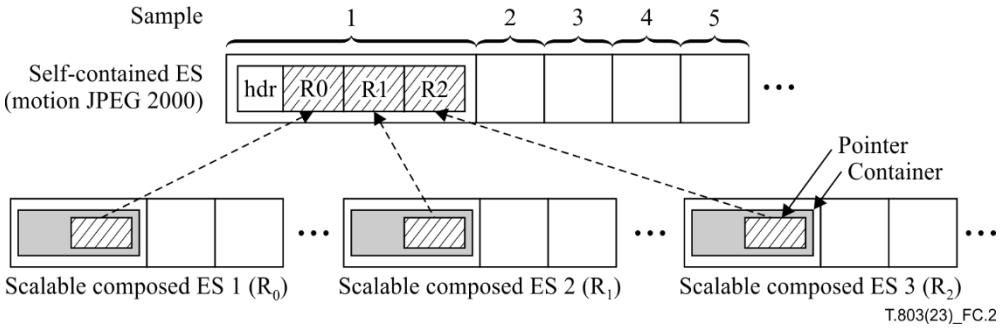


Figure C.2 – Self-contained ES and scalable composed ES

Figure C.3 illustrates the relationship between self-contained ES and decodable composed ESs. In decodable composed ES, a sample is wrapped by a container, which includes the *Data* structure (with header information), and one or more pointer structures referring to data segments in the self-contained ES.

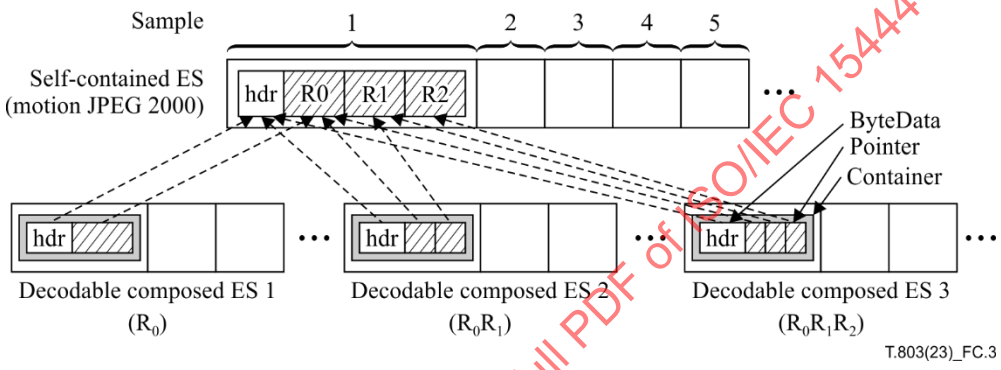


Figure C.3 – Self-contained ES and decodable composed ES

C.4.2 In-stream structures

This clause defines the in-stream structures used in scalable composed ES and decodable composed ES. A composed ES can include media from other elementary streams either by copy or by reference. The ByteData structure is used to include media data by copy while the pointer structure is used to include media data by reference. In addition, each sample is wrapped by a container structure, which can contain one or more ByteData or pointer structures.

```
class aligned(8) DataUnit(type) {
      unsigned int(32)      type;
      unsigned int(32)      size;
      unsigned int(32)      RDHints;
}

class ByteData(type='bdat') extends DataUnit() {
      unsigned int(8) data[];
}

class Container(type='cont') extends DataUnit() {
      DataUnit(type)  units[];
}

class Pointer(type='poin') extends DataUnit() {

      unsigned int(8)       rack_ref_index;
      unsigned int(8) segment_count;
      for(int i=0; i<segment_count; i++) {
            unsigned int(32)      offset;
            unsigned int(32)      length;
      }
}
```

Semantics:

`type` indicates the type of the data structure, like "ByteData", "Container" and "Pointer".

`size` is the size of this structure, including itself and subsequent data in this structure.

`RDHints` indicates the relative or absolute importance of media data contained or referenced by this object. For instance, `RDHints` could be an associated distortion increment value, the amount by which the media distortion will increase if the media data is lost, or it could be importance ranking.

`data[]` is a byte array containing media data in its original format. This Recommendation | International Standard does not specify the format of the media data.

`units[]` is a list of objects contained by container object. The object can be either "ByteData" or "Pointer", but cannot be another "Container" object.

`track_ref_index` specifies the index of the track to which this "pointer" is pointing.

`segment_count` is the number of "offset" and "length" pair.

`offset` is the offset of the first byte within the referred sample to copy.

`length` is the number of bytes of the data segment.

C.5 Protection at file format level

C.5.1 Overview

C.5.2 to C.5.4 describe how media data is protected using the protection schemes (i.e., authentication or description) and how the protection is signalled using the boxes defined in C.3.4 and C.3.5.

When the security tools applied change the data length, it shall update all pointers and length fields in all boxes, to ensure correct parsing by the reader.

C.5.2 Item-based protection for ISO base file format and JPEG family file formats

This annex defines two types of **`ProtectionSchemeInfoBox`**() : authentication and decryption. The two protection schemes protect the scalable media without loss of scalability.

Each instance of a protection scheme and used parameters (like MAC, IV, keys, etc.) are described by a ProtectionSchemeInfoBox located inside ItemProtectionBox. On the other hand, the ItemLocationBox contains a list of items and each item points to one or more contiguous segments of media data. The ItemInformationBox maintains a mapping table between the items in ItemLocationBox and the protection schemes in ItemProtectionBox. Figure C.4 gives an example with two items and three protection schemes: item 0 is protected with Scheme 0 and Scheme 2, while Item 1 is protected with Scheme 1.

As illustrated in Figure C.4, the same item can be applied with multiple protection schemes, e.g., item 0. In addition, the items can overlap with each other, and the overlapped region can be applied with multiple protection schemes. Therefore, it is important to specify the processing order among the protection schemes. The '`ffsc`' file format mandates that the file shall be un-protected in the same order as the schemes appear in '`iinf`' box. For instance, for un-protection, the first scheme appearing in '`iinf`' box is applied first and the last scheme appearing in '`iinf`' box is applied last. In this example, the file is first un-protected with Scheme 2 on Item 0, followed by Scheme 1 on Item 1 and scheme 0 on Item 0.

NOTE – The protection order (with multiple protection tools) is the reverse of the un-protection order, that is, the first scheme appearing in "`iinf`" box is the last tool to be applied, and the last scheme appearing in "`iinf`" box is the first tool to be applied.

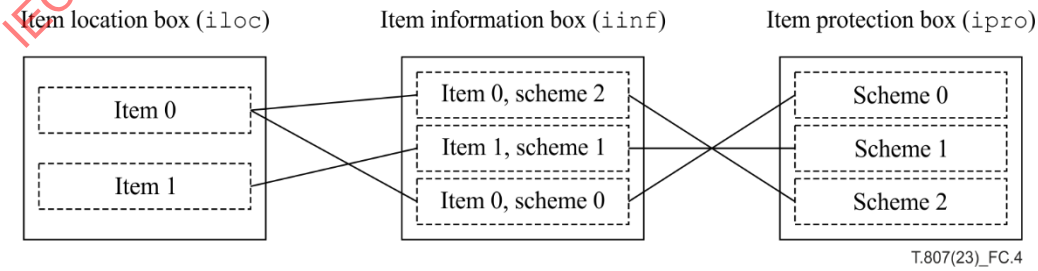


Figure C.4 – Relationship between iloc, iinf and ipro

C.5.3 Additional requirements for item-based protection for JPEG family file formats

JP2, JPX and JPM files are not based on the ISO base file format but have some common boxes, e.g., the File Type Box ('`ftyp`'). In order to use the Item-based protection in this Recommendation | International Standard, these file formats

shall incorporate protection by adding the 'meta', 'hdlr', 'ipro', 'iloc', and 'iinf' from ISO/IEC 14496-12 in addition to the new boxes defined by this Recommendation | International Standard.

Operation of protection is the same as for the ISO base file format. However, codestreams in JPEG family file formats can appear in contiguous codestream boxes in addition to Media Data ('mdat') boxes.

C.5.4 Sample-based protection

When a protection scheme is applied on a sample basis, it is signalled by a ProtectionSchemeInfoBox located in either the ScalableSampleDescriptionEntry or the ScalableSampleGroupEntry. When the ProtectionSchemeInfoBox is located in the ScalableSampleDescriptionEntry, the protection is applied to all samples in the track; when the ProtectionSchemeInfoBox is located in the ScalableSampleGroupEntry, the protection is applied only to the samples in the sample group.

When samples are applied with multiple protection schemes, the ScalableSampleDescriptionEntry or ScalableSampleGroupEntry contains multiple ProtectionSchemeInfoBoxes. Under the major brand of 'ffsc', the samples shall be un-protected in the order that the corresponding ProtectionSchemeInfoBoxes were defined.

Figure C.5 gives an example of "mp4v" sample description entry that is protected with an authentication followed by an encryption. Note that there are two ProtectionSchemeInfoBoxes in this entry: the first one is for the decryption scheme while the second one is for the authentication scheme. To un-protect the sample, the file reader has to apply the decryption scheme followed by the authentication scheme, which is the order in which it appears in the sample entry.

When sample-based protection is applied to composed ES, the protection is actually applied to the media data which is contained or pointed by the container, ByteData or pointers object. For instance, to encrypt a sample in a composed ES, which is wrapped by a container object, the protection process shall encrypt its media data only, retaining the structure of the container.

The sample-based protection can be applied to all samples in a track or sample group as a whole (when GL = 0 in SchemeInformationBox) or separately to each sample (when GL = 1 in SchemeInformationBox). In the former case, the ValueList has only one MAC or IV, and in the latter case, the ValueList has one MAC or IV for each sample.

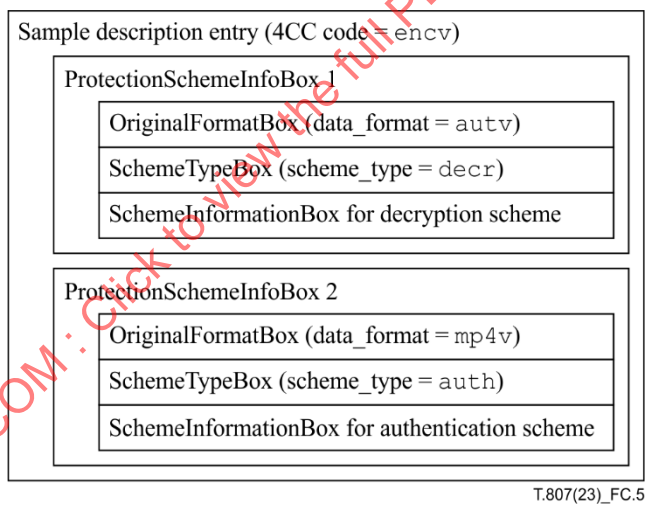


Figure C.5 – An example sample description entry protected by authentication scheme followed by description scheme

C.6 Examples (Informative)

C.6.1 Example 1

This example shows a very simple JPEG file (using JP2 file format) using only authentication. As shown in Figure C.6, authentication is applied to the coded media data stored in the "Contiguous codestream box", authentication algorithm is HMAC with SHA-1 hashing.

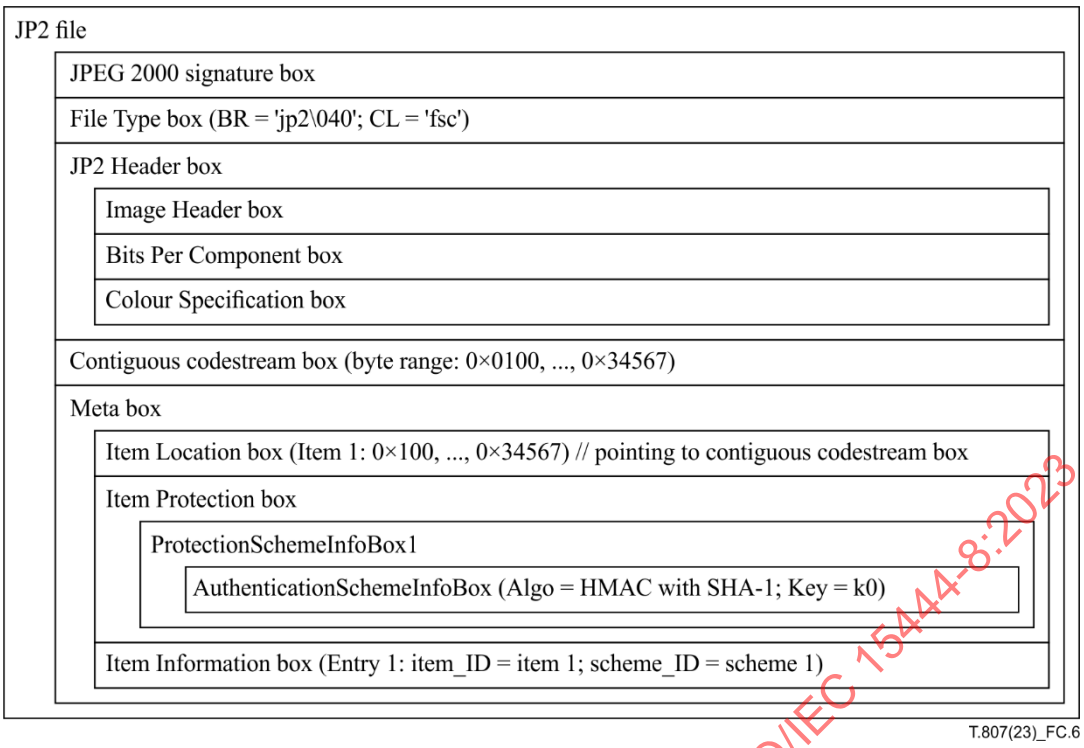


Figure C.6 – Example 1: Item-based protection of JP2 file (authentication)

C.6.2 Example 2

This example corresponds to the example given in 6.3.1. An image is coded with JPEG 2000 and has three resolutions. The first resolution is not encrypted in order to provide preview capability, and the second and third resolutions are encrypted with keys k1 and k2, respectively. The input image is coded in RLCP progression order and has 1 tile and 3 resolutions. (The number of layers, components and precincts are not important in this specific example.) Encryption is performed using AES in CBC mode without padding (using cipher-text stealing), using k1 to encrypt resolution 1 and using k2 to encrypt resolution 2, and resolution 0 is left unencrypted.

First of all, the ItemLocationBox contains two Items: one item points to the byte range from 0x31CC to 0xA3E8 and the other one points to the byte range from 0xA3E9 to 0x31101. The ItemProtectionBox contains two DecryptionSchemeInformationBoxes: the first one uses AES in CBC mode and k1, the other uses AES in CBC mode and k2. The ItemInformationBox links the DecryptionSchemeInformationBoxes in ItemProtectionBox to the Items in ItemLocationBox. In ItemDescriptionBox, ItemDescription 1 describes Item 1 as resolution 1 of the image, while ItemDescription 2 describes Item 2 as resolution 2 of the image.

Note that JPEG 2000 image can be located in either MDAT box in the sample file as the META box, or in a different file whose format is described by the ISO base file format, as shown in Figure C.7.

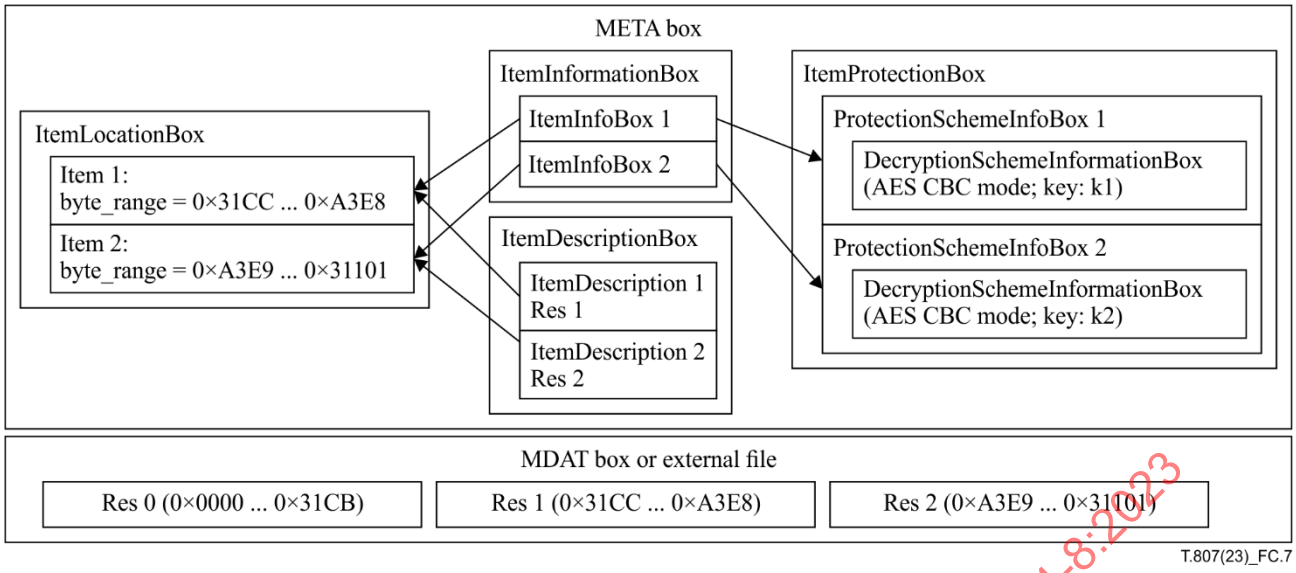


Figure C.7 – Example 2: Item-based protection of a JPEG 2000 images (encryption)

C.6.2.1 Transcoding to resolution 1

To securely transcode the above example to lower resolution by discarding resolution 2, the transcoder needs to do the following:

- Discard the trunk of media data corresponding to resolution 2, i.e., byte range 0xA3E9 – 0x31101.
- Remove Item 2 from ItemLocationBox.
- Remove ItemInfoBox 2 from ItemInformationBox.
- Remove ItemDescription 2 from ItemDescriptionBox.
- Remove ProtectionSchemeInfoBox 2 from ItemProtectionBox.

After transcoding, the resulting File format is shown in Figure C.8. Note that the transcoder does not have to decrypt the media data, and thereby achieve end-to-end security.

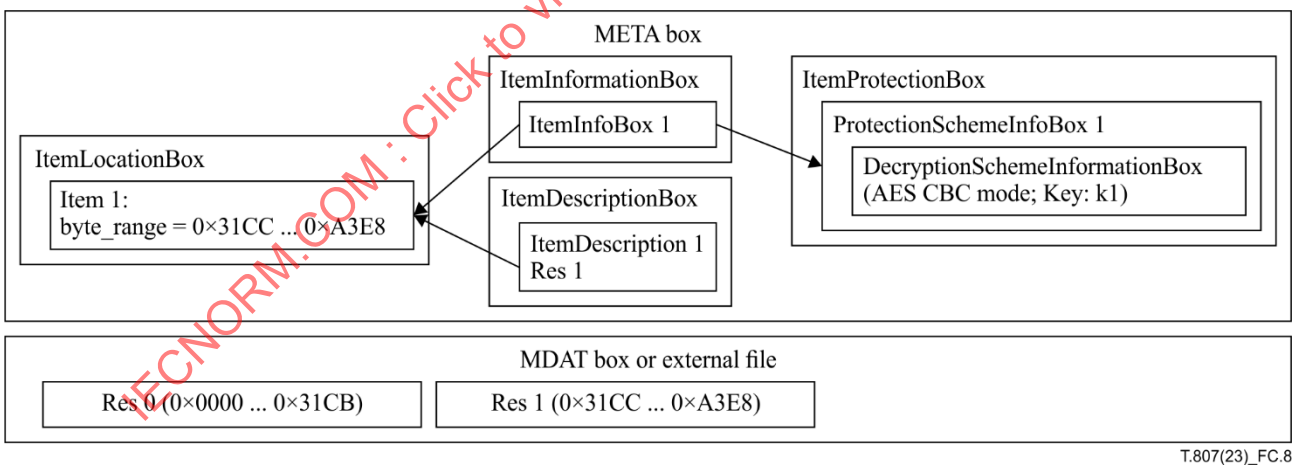


Figure C.8 – Example 2: Secure transcoding to lower resolution (discarding resolution 2)

C.6.3 Example 3

This example corresponds to the example given in 6.3.2. In this case, authentication is applied to the same JPEG 2000 coded image as in Example 1. In this example all three resolutions are authenticated, where the authentication for each resolution uses a different key. Each resolution is authenticated using different keys, and each layer within a resolution has its own MAC value. In summary, there will be a total of three keys and nine MAC values for the entire image. Specifically

- Resolution 0 has three MAC values M0, M1 and M2 (one for each layer) using K0
- Resolution 1 has three MAC values M3, M4 and M5 (one for each layer) using K1

- Resolution 2 has three MAC values M6, M7 and M8 (one for each layer) using K2

The authentication is performed using HMAC with SHA-1 hash. As shown in Figure C.9, the ItemLocationBox has three items corresponding to three resolutions, and each item has three extents corresponding to three layers within a resolution. The ItemProtectionBox contains three ProtectionSchemeInformationBoxes, the first one signals the applied authentication tool using K0 and three MAC values M0, M1 and M2, and so forth. The ItemInformationBox and ItemDescriptionBox are used in the same way as the previous example.

Note that "granularity" field in the AuthenticationSchemeInformationBox is used to indicate the smaller protection unit of the authentication tool. When "granularity" is "extent", the authentication tool will generate one MAC for each extent (i.e., three MAC values for the entire Item in this example); when it is set to "item", it will generate only one MAC value for the entire item.

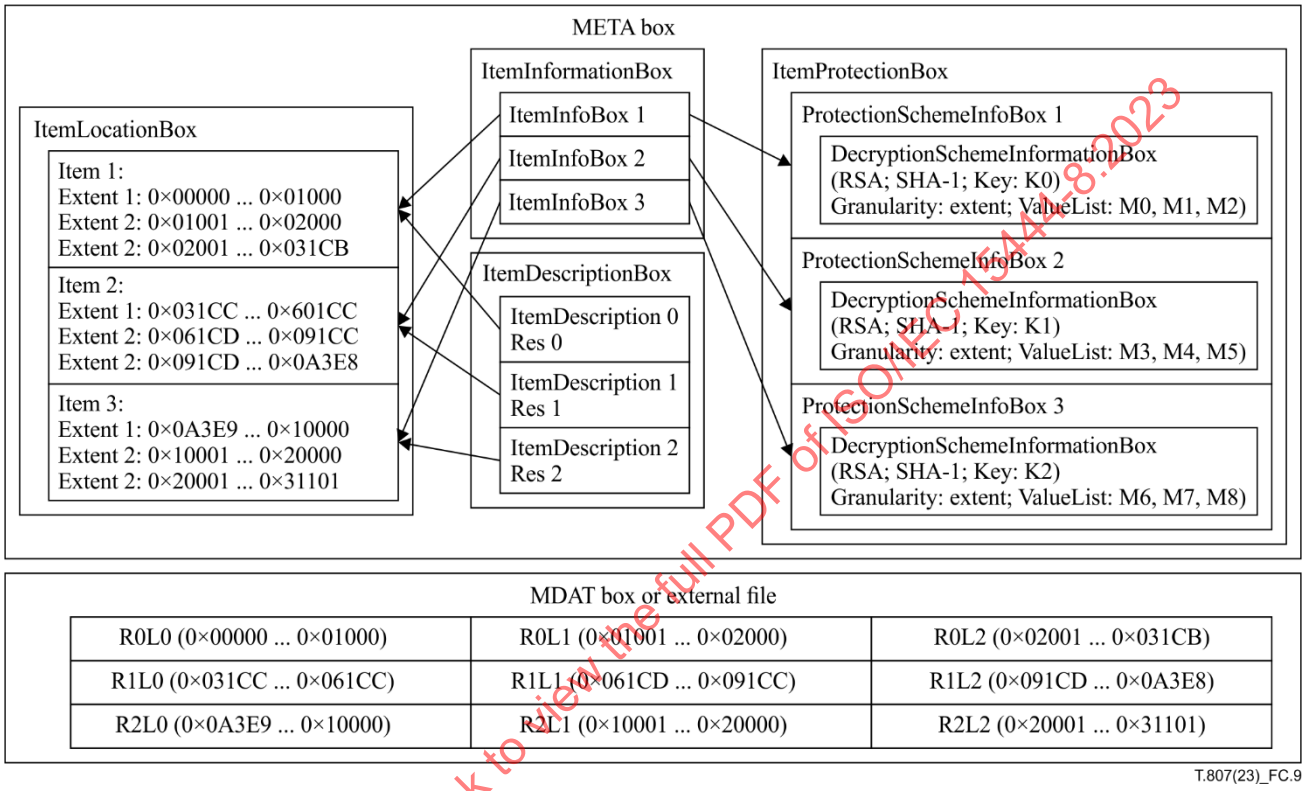


Figure C.9 – Example 3: Item-based protection of a JPEG 2000 image (Authentication)

C.6.3.1 Transcoding to resolution 1

To securely transcode the above example to resolution 1, the transcoding has to do the following:

Discard trunk of media data corresponding to resolution 1, i.e., byte range 0xA3E9 – 0x31101.

- Remove Item 3 from ItemLocationBox
- Remove ItemInfoBox 3 from ItemInformationBox
- Remove ItemDescription 3 from ItemDescriptionBox
- Remove ProtectionSchemeInfoBox 3 from ItemProtectionBox

The resulting file format of the transcoded codestream is shown in Figure C.10.

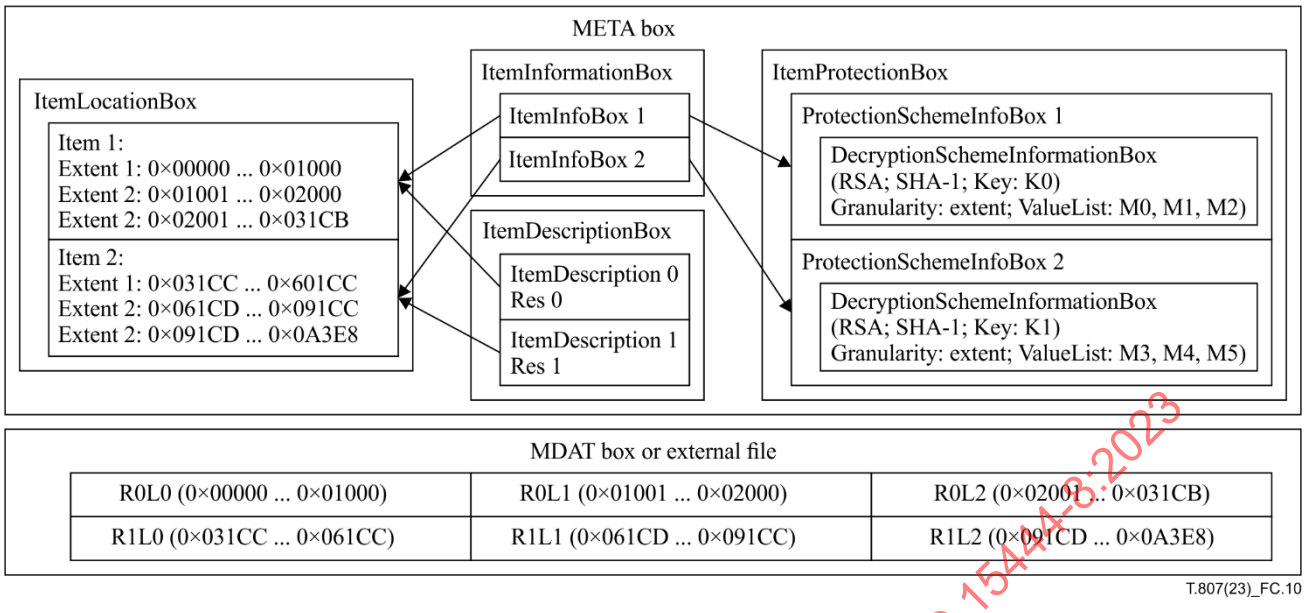


Figure C.10 – Example 3: Transcoding to resolution 1

C.6.4 Example 4

This example illustrates sample-based protection for a time-sequenced JPEG 2000 coded pictures. Each picture has three layers (L0, L1 and L2), and all three layers are authenticated using a unique key (AK0, AK1 and AK2). After that, the L1 and L2 are encrypted using a unique key (CK0 and CK1). That is, L0 is authenticated with AK0 producing one MAC value M0. L1 and L2 are first authenticated with AK1 and AK2 (producing M1 and M2) and then encrypted using CK0 and CK1 respectively. Authentication is applied using HMAC with SHA-1 and encryption is achieved using AES in CBC mode.

To preserve scalability at the file format level, three scalable composed ESs are generated, ES0, ES1, and ES2, one for each layer. Each ES is described by one media track, for instance, ES0 is described by Trak0, ES1 is described by Trak1, and so on.

The format of the three tracks (Trak 0, Trak 1 and Trak 2) is illustrated in Figure C.11. Trak 0 has one ProtectionSchemeInfoBox as L0 is protected by authentication tool only. Trak 1 and Trak 2 have two ProtectionSchemeInfoBox as L1 and L2 are protected by both authentication and encryption tool. Note that the decryption tool appears in the first ProtectionSchemeInfoBox and the authentication tool appears in the second ProtectionSchemeInfoBox. The un-protection process has to follow the same order to get decodable JPEG 2000 codestream.

Note that for decodable composed ES and self-contained ES, sample-based protection is applied in the same way as for scalable composed ES.

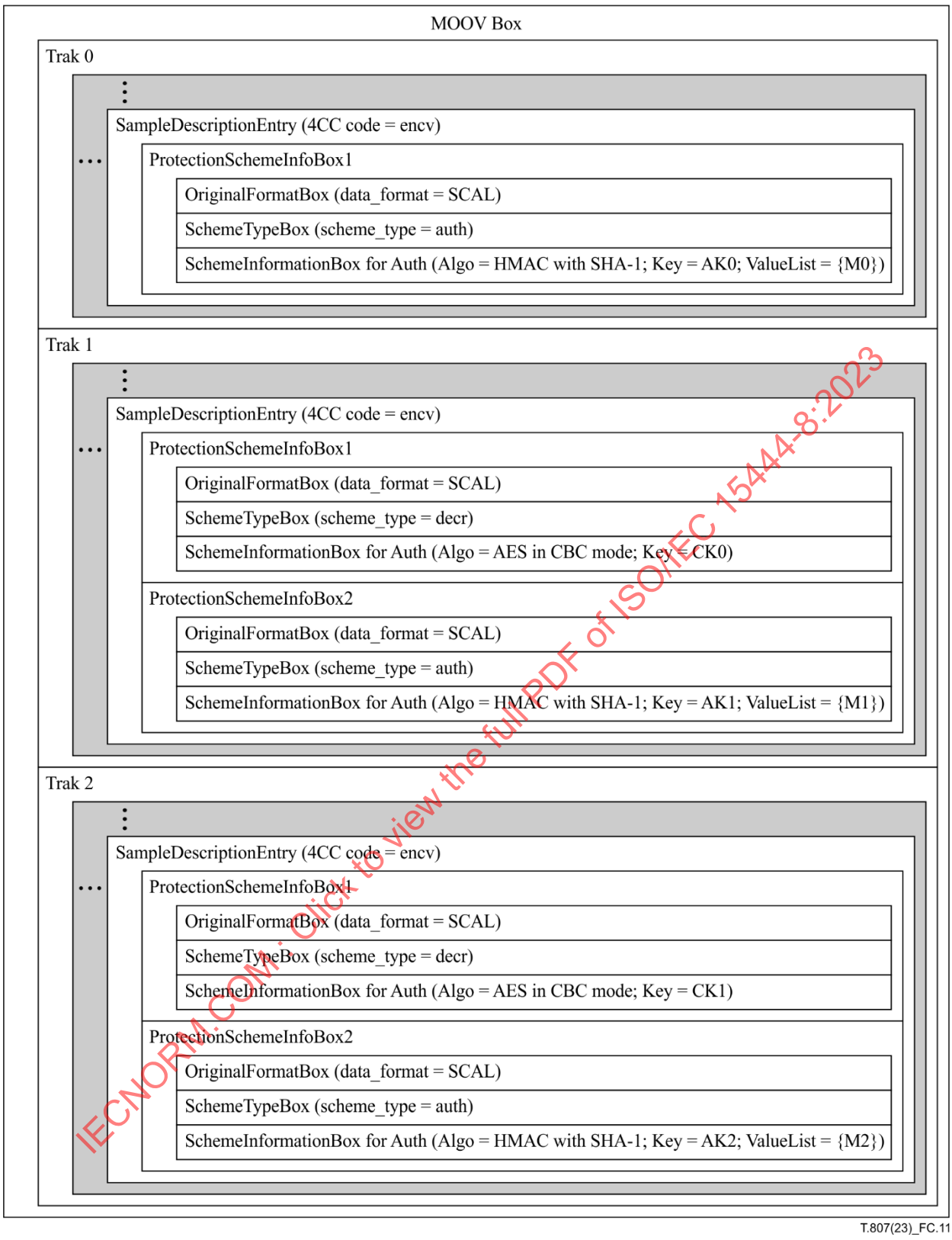


Figure C.11 – Example 4: Sample-based protection of a time-sequenced JPEG 2000 pictures

C.6.4.1 Transcoding to layer 1

To securely transcode the sequence of JPEG 2000 picture to layer 1, the transcoder needs to do the following:

- Discard the scalable composed ES 2 corresponding to layer 2, and discard the trunk of media data in self-contained ES.
- If necessary, the transcoder also needs to update the byte range values of Pointer objects in the remaining two scalable composed ESs.

- Discard Trak 2 that describes scalable composed ES 2.

The resulting file format is shown in Figure C.12.

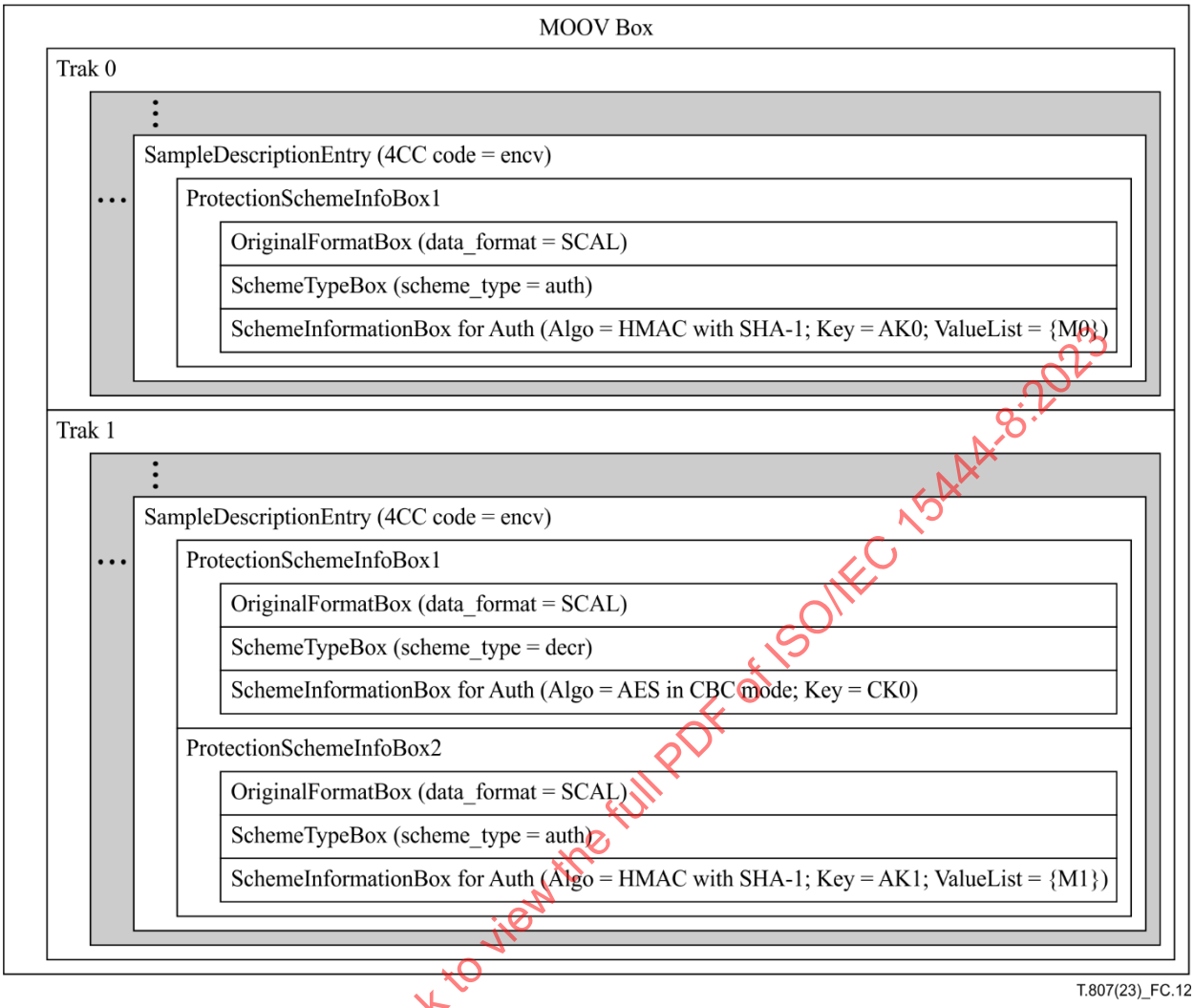


Figure C.12 – Example 4: Secure transcoding to lower SNR quality (layer 1)

C.6.5 Example 5

This example illustrates how sample-based protection is applied for video browsing and video summarization. In this example, a sequence of 10000 frames is summarized into 10 scenes. For instance, the first 1000 frames constitute the first scene; the second 1000 frames constitute the second scene, and so on. For each scene, the first 50 frames (5-second video at 10 fps) are left unencrypted for preview purposes, the rest of the frames are encrypted using AES in CBC mode with key K0. In this example, the 10000 pictures are stored in a self-contained ES, which can be located in MDAT box or external file whose format is not specified by the ISO base file format.

Figure C.13 illustrates the file format when the above protection is applied to a self-contained ES, which is described by Track 0. The SampleToGroupBox has 20 entries, the first 50 pictures of every scene are mapped to no descriptor and the next 950 pictures of each scene are mapped to the first ScalableSampleGroupEntry in SampleGroupDescriptionBox. The grouping_type is "prot", for protection purposes.

The ScalableSampleGroupEntry has handler_type of "encv" and the protection is applied to all resolutions (res = –1), all layers (layer = –1) and all regions of each sample (`cropped_width=cropped_height=0`). As only one protection is applied to this sample group, the ScalableSampleGroupEntry contains only one ProtectionSchemeInfoBox, which in turn contains the OriginalFormatBox, the SchemeTypeBox and the SchemeInformationBox.

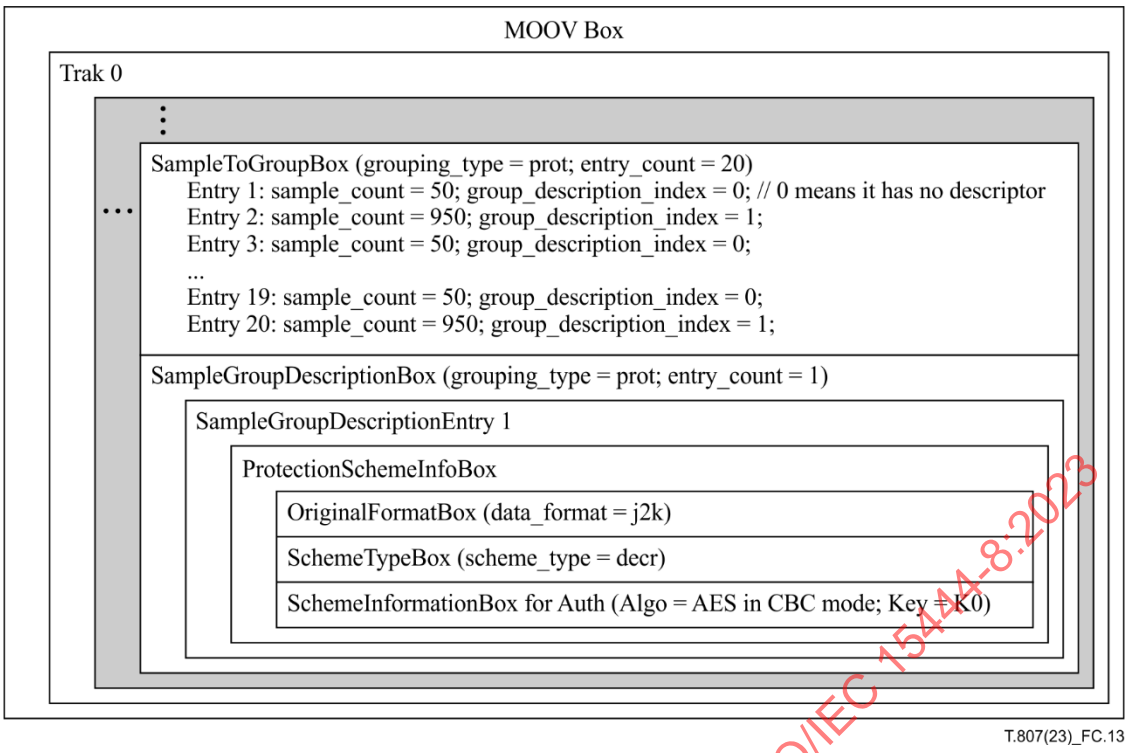


Figure C.13 – Example 5: Sample-based protection for video browsing or video summarization

C.6.5.1 Example 5: Transcoding to shorter time length

This example illustrates how to securely transcode the above codestream to shorter the time length, i.e., discarding the last 5000 pictures. The transcoder has to do the following things:

- Discard the trunk of media data corresponding to the last 5000 pictures.
- Remove the entries corresponding to the last 5000 pictures in SampleToGroupBox.

The resulting codestream is shown in Figure C.14.

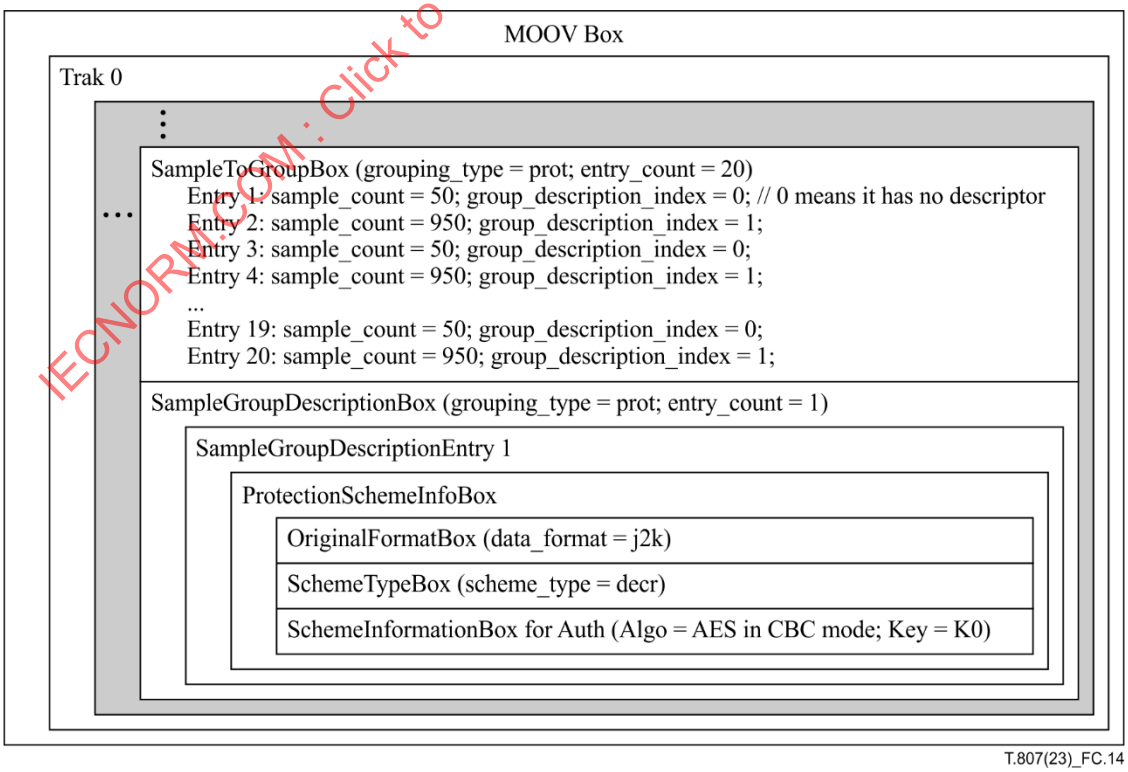


Figure C.14 – Example 5: Transcoding to shorter time length (discarding the last 5000 pictures)

C.6.6 Example 6

This example demonstrates how the structures defined in this annex can be used to perform authentication and decoding of verified data, as shown in Figure C.15.

An authentication adaptor/transcoder removes data that is not verifiable with the available media data and authentication data. For example, in a streaming system, some media packets can be lost during transmission. A file format receiver can reconstruct the received data to the best of its ability based on the available data. Then, an authentication adaptor/transcoder can determine which data can be verified, and then remove the packets that are not verifiable. The resulting file only contains the decodable, verified data.

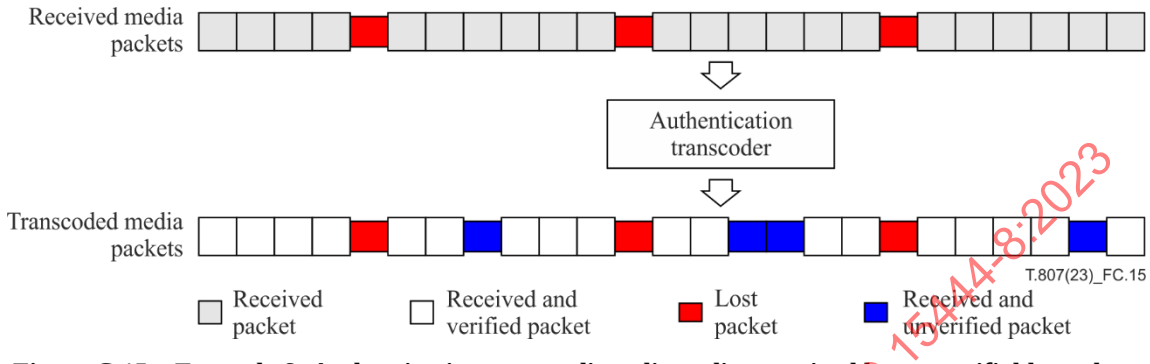


Figure C.15 – Example 6: Authentication transcoding, discarding received but unverifiable packets

C.6.7 Example 7

This example shows the effect of length changes on the contents of a file due to protection coding.

The 'moov' box expanded in Figure C.16 contains boxes that reference samples in the 'mdat' box. In this case the 'mdat' box contains six samples, labelled ED1 to ED6.

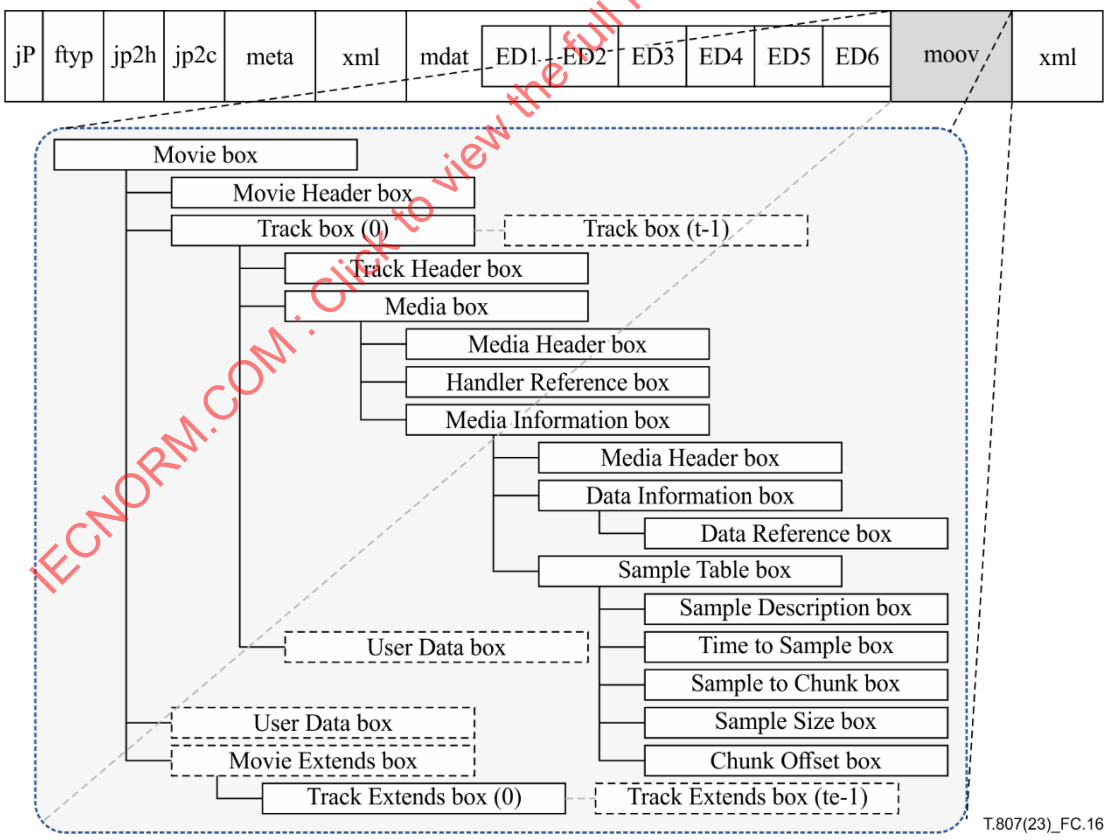


Figure C.16 – Motion JPEG 2000 file with detailed box structure

Without any protection, some of the boxes and the references to those boxes appear as shown in Figure C.17.

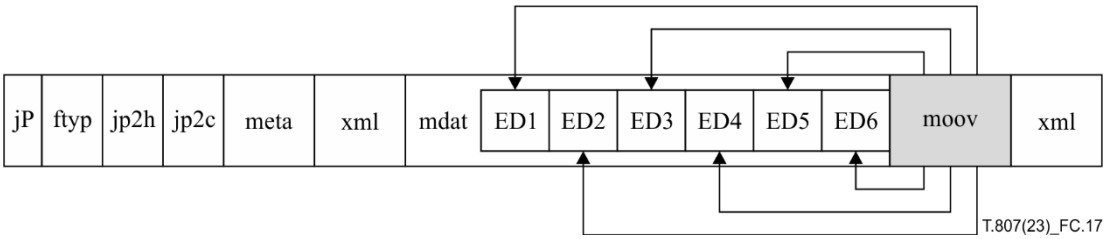


Figure C.17 – Simplified Motion JPEG 2000 box structure showing references

Figure C.18 shows boxes or samples in pink that have been protected and can have had their length changed. The pointers in the 'moov' box have been adjusted to point to the correct location of the protected samples.

NOTE – In order to function as shown in those figures, the protection tool understands all pointers present in the file. Thus, a protected Motion JPEG 2000 file requires a protection tool capable of adjusting Motion JPEG 2000 references. The alternative is for the file with portions that have been changed to be marked as unreadable by a Motion JPEG 2000 reader until all length changing operations have been undone.

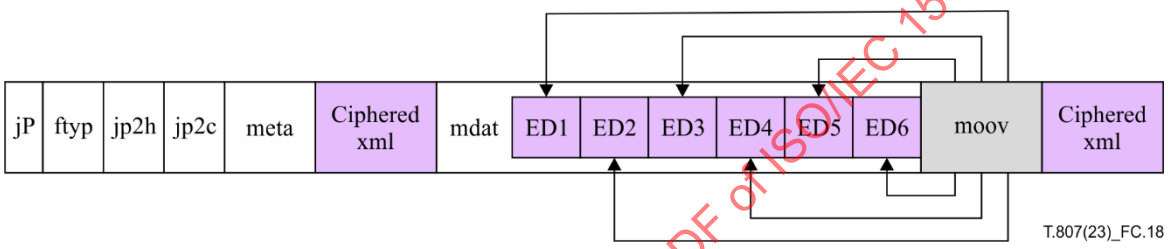


Figure C.18 – Simplified Motion JPEG 2000 box structure showing references after length changing protection operations

C.6.8 Example 8

These examples in Figures C.19, C.20 and C.21 show the effect of length changes on the contents of a JPM file due to protection coding.

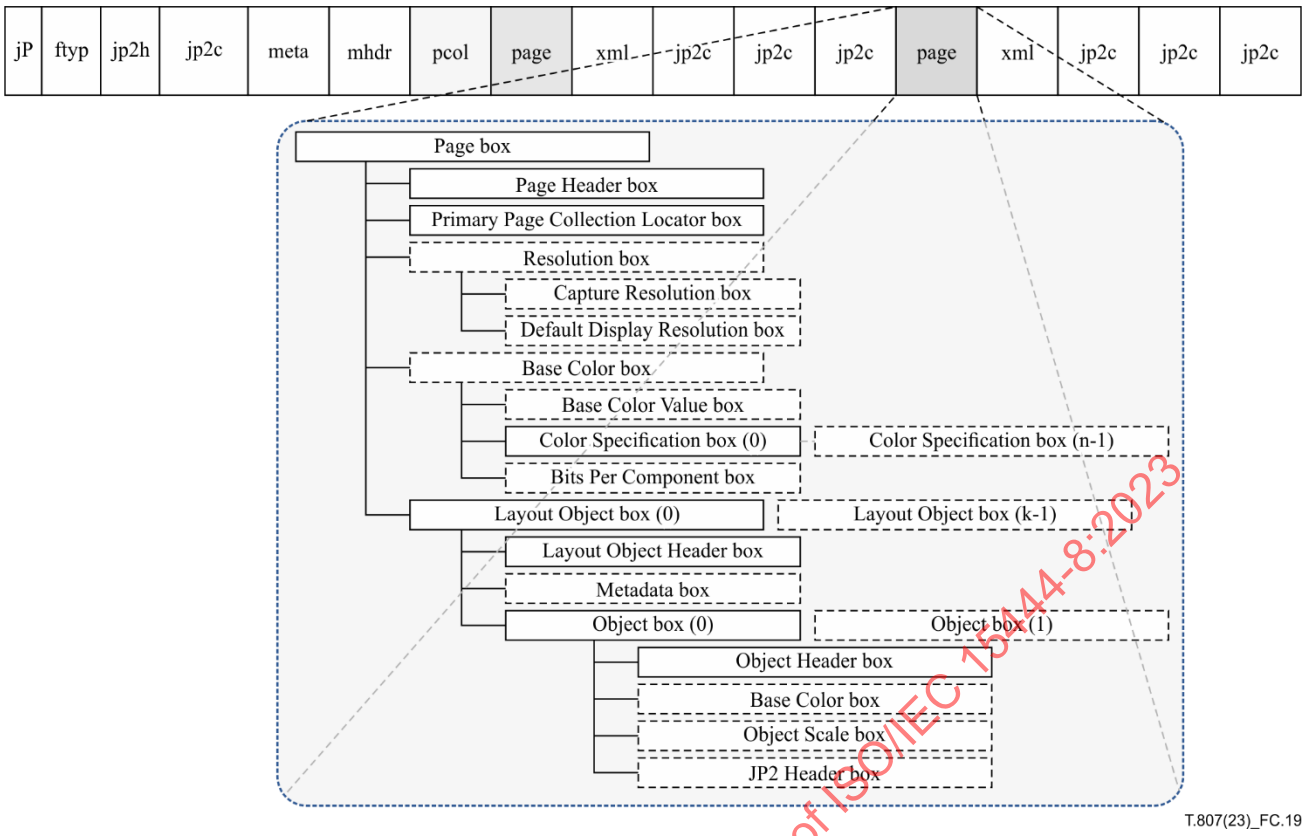


Figure C.19 – JPM file with detailed box structure

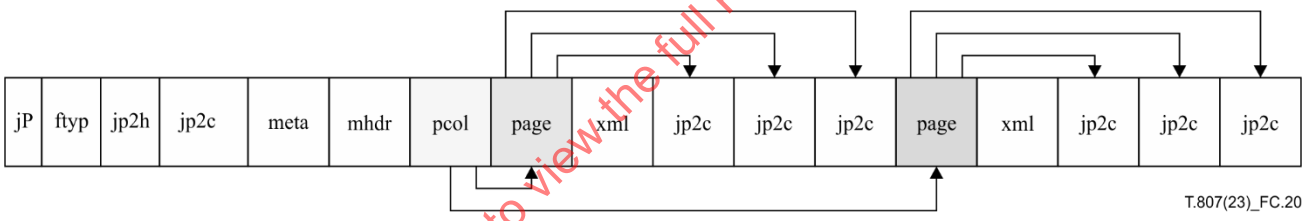


Figure C.20 – Simplified JPM box structure showing references

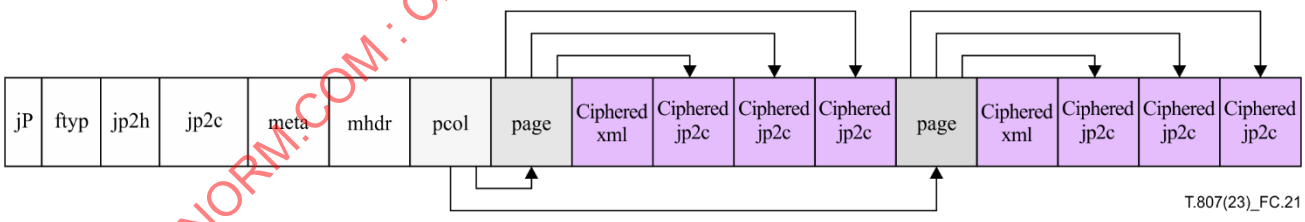


Figure C.21 – JPM box structure showing references after length changing protection operations

C.7 Boxes defined in ISO/IEC 14496-12 (informative)

This clause is not a normative part of this Recommendation | International Standard. The definitions listed here are repeated from ISO/IEC 14496-12 for convenience.

Box

Files are formed as a series of objects, called boxes in this Recommendation | International Standard. All data are contained in boxes; there are no other data within the file.

Boxes start with a header which gives both size and type. The header permits compact or extended size (32 or 64 bits) and compact or extended types (32 bits or full UUIDs). The standard boxes all use compact types (32-bit) and most boxes will use the compact (32-bit) size. Typically only the Media Data Box(es) need the 64-bit size.

The size is the entire size of the box, including the size and type header, fields, and all contained boxes. This facilitates general parsing of the file. The definitions of boxes are given in the syntax description language (SDL) defined in MPEG-4 (see reference in clause 2). Comments in the code fragments in this Recommendation | International Standard indicate informative material. The fields in the objects are stored with the most significant byte first, commonly known as network byte order or big-endian format.

```
aligned(8) class Box(unsigned int(32) boxtype, optional unsigned int(8)[16] extype) {
      unsigned int(32)      size;
      unsigned int(32)      type = boxtype;
      if (size==1) {
            unsigned int(64)      argesize;
      } else if (size == 0) {
            // box extends to end of file
      }
      if (boxtype == 'uuid') {
            unsigned int(8)[16]   usertype = extype;
      }
}
```

The semantics of these two fields are:

`size` is an integer that specifies the number of bytes in this box, including all its fields and contained boxes; if size is 1 then the actual size is in the field large size; if size is 0, then this box is the last one in the file, and its contents extend to the end of the file (normally only used for a Media Data Box).

`type` identifies the box type; standard boxes use a compact type, which is normally four printable characters, to permit ease of identification, as reflected below. User extensions use an extended type; in this case, the type field is set to 'uuid'.

Boxes with an unrecognized type shall be ignored and skipped.

Many objects also contain a version number and a flag field:

```
aligned(8) class FullBox(unsigned int(32) boxtype, unsigned int(8) v, bit(24) f)
extends Box(boxtype) {
      unsigned int(8) version = v;
      bit(24)    flags = f;
}
```

The semantics of these two fields are:

`version` is an integer that specifies the version of this format of the box.

`flags` is a map of flags.

Boxes with an unrecognized version shall be ignored and skipped.

Item Location Box

Definition

Box Type: 'iloc'

Container: Meta box ('meta')

Mandatory: No

Quantity: Zero or one

The item location box provides a directory of resources in this or other files, by locating their containing file, their offset within that file, and their length. Placing this in binary format enables common handling of this data, even by systems which do not understand the particular metadata system (handler) used. For example, a system might integrate all the externally referenced metadata resources into one file, re-adjusting file offsets and file references accordingly.

The box starts with three values, specifying the size in bytes of the offset field, length field, and `base_offset` field, respectively. These values shall be from the set {0, 4, 8}.

Items can be stored fragmented into extents, e.g., to enable interleaving. An extent is a contiguous subset of the bytes of the resource; the resource is formed by concatenating the extents. If only one extent is used (`extent_count = 1`), then either or both of the offset and length can be implied:

- If the offset is not identified (the field has a length of zero), then the beginning of the file (offset 0) is implied.
- If the length is not specified, or specified as zero, then the entire file length is implied. References into the same file as this metadata, or items divided into more than one extent, should have an explicit offset and length, or use a MIME type requiring a different interpretation of the file, to avoid infinite recursion.

The size of the `item` is the sum of the `extent_length(s)`.

NOTE – Extents can be interleaved with the chunks defined by the sample tables of tracks.

The data-reference index can take the value 0, indicating a reference into the same file as this metadata, or an index into the data-reference table.

Some referenced data can itself use offset/length techniques to address resources within it (e.g., an MP4 file might be 'included' in this way). Normally such offsets are relative to the beginning of the containing file. The field 'base offset' provides an additional offset for offset calculations within that contained data. For example, if an MP4 file is included within a file formatted to this Recommendation | International Standard, then normally data-offsets within that MP4 section are relative to the beginning of file; `base_offset` adds to those offsets.

Syntax

```
aligned(8) class ItemLocationBox() extends FullBox('iloc', version = 0, 0) {
      unsigned int(4) offset_size;
      unsigned int(4) length_size;
      unsigned int(4) base_offset_size;
      unsigned int(4) reserved;
      unsigned int(16)     item_count;
      for (i=0; i < item_count; i++) {
            unsigned int(16)     item_ID;
            unsigned int(16)     data_reference_index;
            unsigned int(base_offset_size*8)     base_offset;
            unsigned int(16)     extent_count;
            for (j=0; j < extent_count; j++) {
                  unsigned int(offset_size*8)     extent_offset;
                  unsigned int(length_size*8)     extent_length;
            }
      }
}
```

Semantics

`offset_size` is taken from the set {0, 4, 8} and indicates the length in bytes of the offset field.

`length_size` is taken from the set {0, 4, 8} and indicates the length in bytes of the length field.

`base_offset_size` is taken from the set {0, 4, 8} and indicates the length in bytes of the `base_offset` field.

`item_count` counts the number of resources in the following array.

`item_ID` is an arbitrary integer 'name' for this resource which can be used to refer to it (e.g., in a URL).

`Data_reference_index` is either zero ('this file') or a 1-based index into the data references in the data information box.

`base_offset` provides a base value for offset calculations within the referenced data. If `base_offset_size` is 0, `base_offset` takes the value 0, i.e., it is unused.

`extent_count` provides the count of the number of extents into which the resource is fragmented; it shall have the value 1 or greater.

`extent_offset` provides the absolute offset in bytes from the beginning of the containing file, of this item. If `offset_size` is 0, offset takes the value 0.

`extent_length` provides the absolute length in bytes of this metadata item. If `length_size` is 0, length takes the value 0. If the value is 0, then length of the item is the length of the entire referenced file.

Item Information Box

Definition

Box Type: '`iinf`'

Container: Meta Box ('meta')

Mandatory: No

Quantity: Zero or one

The Item information box provides extra information about selected items, including symbolic ('file') names. It can optionally occur, but if it does, it shall be interpreted, as item protection or content encoding can have changed the format of the data in the item. If both content encoding and protection are indicated for an item, a reader should first un-protect the item, and then decode the item's content encoding. If more control is needed, an IPMP sequence code can be used.

This box contains an array of entries, and each entry is formatted as a box. This array is sorted by increasing item_ID in the entry records.

Syntax

```
aligned(8) class ItemInfoEntry() extends FullBox('infe', version = 0, 0) {
      unsigned int(16)      item_ID;
      unsigned int(16)      item_protection_index
      string          item_name;
      string          content_type;
      string          content_encoding; //optional
}

aligned(8) class ItemInfoBox() extends FullBox('iinf', version = 0, 0) {
      unsigned int(16)      entry_count;
      ItemInfoEntry() item_infos[entry_count];
}
```

Semantics

item_id contains either 0 for the primary resource (e.g., the XML contained in an 'xml' box) or the ID of the item for which the following information is defined.

item_protection_index contains either 0 for an unprotected item, or the one-based index into the item protection box defining the protection applied to this item (the first box in the item protection box has the index 1).

item_name is a null-terminated string in UTF-8 characters containing a symbolic name of the item.

content_type is the MIME type for the item.

content_encoding is an optional null-terminated string in UTF-8 characters used to indicate that the binary file is encoded and needs to be decoded before being interpreted. The values are as defined for Content-Encoding for HTTP/1.1. Some possible values are "gzip", "compress" and "deflate". An empty string indicates no content encoding.

entry_count provides a count of the number of entries in the following array.

Item Protection Box

Definition

Box Type: 'ipro'

Container: Meta box ('meta')

Mandatory: No

Quantity: Zero or one

The item protection box provides an array of item protection information, for use by the Item Information Box.

Syntax

```
aligned(8) class ItemProtectionBox() extends FullBox('ipro', version = 0, 0) {
      unsigned int(16)      protection_count;
      for (i=1; i <= protection_count; i++) {
            ProtectionSchemeInfoBox() protection_information;
      }
}
```

Semantics

`protection_count` is the number of protection tools applied.

`protection_information` contains the information for each protection tool applied.

Annex D

Technology examples

(This annex does not form an integral part of this Recommendation | International Standard.)

D.1 Introduction

The JPSEC syntax allows normative and non-normative security tools to be applied to JPEG 2000 images. D.2 to D.11 describe ten informative technology examples that demonstrate different usages of JPSEC. These examples are purely informative and not endorsed by this Recommendation | International Standard. However, they are provided to demonstrate the flexibility of the standard.

The technology examples include:

- A flexible access control scheme for JPEG 2000;
- A unified authentication framework for JPEG 2000 images;
- A simple packet-based encryption method for JPEG 2000 codestreams;
- Encryption tool for JPEG 2000 access control;
- Key generation tool for JPEG 2000 access control;
- Wavelet and bitstream domain scrambling for conditional access control;
- Progressive access for JPEG 2000 codestream;
- Scalable authenticity of JPEG 2000 codestreams;
- JPEG 2000 data confidentiality and access control system based on data splitting and luring;
- Secure scalable streaming and secure transcoding.

D.2 A flexible access control scheme for JPEG 2000 codestreams

D.2.1 Security service

An access control scheme allows for rendering JPEG 2000 codestreams according to any combination of resolutions, quality layers, tiles and precincts.

D.2.2 Typical application

It provides protection of content delivery via variable media, e.g., Internet, digital cable TV, satellite broadcast and CD-ROM. Generally, the technology is viable to the applications where a codestream is encrypted only once on the publisher side but the protected codestream is decrypted many ways according to the different privilege on the user sides.

D.2.3 Motivation

In the Super-distribution model, the publisher distributes the protected content freely and the content keys securely. A user who desires to access portions of a codestream sends his/her request to the key server. The key server, in turn, responds with appropriate decryption keys according to the user's privilege. The user can access the allowed sub-images.

D.2.4 Technical overview

A protected JPEG 2000 codestream is produced by encrypting each packet by the publisher. The core of the technology is how to manage a key tree which is constructed in any order of tiles, components, resolutions, layers, precincts, and even code-blocks. To describe the technology easily, assume that the key tree order is RLCP (resolution-layer-component-precinct) and each resolution has the same number of precincts. In the following, given a one-way hash function h(.), consider a JPEG 2000 image codestream with n_T tiles, n_C components, n_L layers, n_R resolution per tile-component, n_P precincts per resolution. With a master key K for a JPEG 2000 codestream. Construct a key tree as follows.

1) Generate key $k^t = h(K\,|"T"|t)$, for each tile $T_{type} = 0, 1, \ldots, n_T - 1$, where "|" is the concatenation, and *"T"* denotes the ASCII code of the letter *T*.

2) Generate key $k^r = h(k^{r+1})$, for each $r = n_R - 2, \ldots, 1, 0$, where $k^{n_R - 1} = h(k^t\,|"R")$ and "*R*" denotes the ASCII code of the letter *R*.

3) Compute key $k^{rl} = h(k^{r(l+1)})$, for each $r = n_R - 1, \ldots, 1, 0$, $l = n_L - 2, \ldots, 1, 0$, where $k^{r(n_L - 1)} = h(k^r|"L")$ and where "*L*" denotes the ASCII code of the letter *L*.

4) Calculate key $k^{rlc} = h(k^{rl}\,|"C"|c)$, for each $r = n_R - 1, \ldots, 1, 0$, $l = n_L - 1, \ldots, 1, 0$, $c = 0, 1, \ldots, n_C - 1$, where "*C*" denotes the ASCII code of the letter *C* and *c* denotes the index of this component.