Multimedia Systems
DOI 10.1007/s00530-008-0150-0

REGULAR PAPER

A survey on JPEG2000 encryption

Dominik Engel · Thomas Stütz · Andreas Uhl

Received: 11 June 2008 / Accepted: 16 December 2008

Abstract Image and video encryption has become a widely discussed topic; especially for the fully featured JPEG2000 compression standard numerous approaches have been proposed. A comprehensive survey of state-of-the-art JPEG2000 encryption is given. JPEG2000 encryption schemes are assessed in terms of security, runtime and compression performance and their suitability for a wide range of application scenarios.

1 Introduction

A clear trend toward the increased employment of JPEG2000 for specialized applications has been observable recently, especially where a high degree of quality or scalability is desired. For example, the Digital Cinema Initiative (DCI), an entity created by seven major motion picture studios, has adopted JPEG2000 as the (video!) compression standard in their specification for a unified Digital Cinema System [8]. With increasing usage comes the increasing need for practical security methods for JPEG2000. Over the last years, a significant number of different encryption schemes for visual data types have been proposed (see [23,60] for extensive overviews). Recently, the awareness for JPEG2000 security has grown with the finalization of part 8 of the JPEG2000 standard, JPSEC [32].

The most secure method for the encryption of visual data, sometimes referred to as the naive method, is to encrypt the whole multimedia stream (e.g., a JPEG2000 codestream) with the aid of a cryptographically strong cipher like AES [7]. The most prominent reasons not to stick to classical full encryption of this type for multimedia applications are

- to maintain format compliance and/or associated functionalities like scalability (which is usually achieved by parsing operations and marker avoidance strategies),
- to achieve higher robustness against channel and storage errors, and
- to reduce the computational effort (which is usually achieved by trading off security, as is the case in partial or soft encryption schemes).

These issues immediately make clear that encryption methods for visual data types need to be specifically tailored to fulfill the requirements of a particular multimedia application with respect to security on the one hand and other functionalities on the other hand.

A number of proposals for JPEG2000 encryption have been put forward to date. The approaches differ significantly in their fields of applications, their levels of security, the functionalities they provide and their computational demands.

In this paper our aim is to give a comprehensive survey of the existing approaches. For this purpose, we first present different categories for the classification of JPEG2000 encryption schemes. We systematically describe, discuss, evaluate, and compare the various techniques, especially with respect to their impact on JPEG2000 compression performance,

Communicated by A.U. Mauthe.

D. Engel · T. Stütz · A. Uhl (✉)
Department of Computer Sciences, Salzburg University,
Jakob-Haringerstr. 2, Salzburg, Austria
e-mail: uhl@cosy.sbg.ac.at

D. Engel
e-mail: dengel@cosy.sbg.ac.at

T. Stütz
e-mail: tstuetz@cosy.sbg.ac.at

Published online: 16 January 2009

concerning their security, and regarding their computational performance.

In Sect. 2 we give an introduction to media encryption. Section 3 provides an overview of the JPEG2000 standard suite, focusing on the parts relevant to our survey, most importantly Part 8, JPSEC. In Sect. 4 we discuss evaluation criteria for JPEG2000 encryption schemes.

In Sect. 5 we cover methods for bitstream-oriented encryption techniques, Sect. 6 is devoted to compression-integrated methods. In Sect. 7 we discuss the findings of this survey and we give recommendations which techniques should preferably be used in specific application scenarios. Section 8 concludes the paper.

2 Media encryption

In the following, we discuss a number of useful categories for the classification of media encryption schemes, which of course are also relevant for JPEG2000 encryption.

2.1 Security and quality constraints

Encryption may have an entirely different aim as opposed to maximal confidentiality or privacy in the context of certain multimedia applications. "Transparent encryption" [41] has been introduced mainly in the context of digital TV broadcasting (also called "perceptual encryption" predominantly in the area of audio encryption): a pay TV broadcaster does not always intend to prevent unauthorized viewers from receiving and watching his program, but rather intends to promote a contract with non-paying watchers. This can be facilitated by providing a low quality version of the broadcast program for everyone, only legitimate (paying) users get access to the full quality visual data (which has been already broadcast together with the low quality version in encrypted form). Also, the degree of confidentiality varies from application to application. Whereas a high degree is required for applications like video conferencing, telemedicine, or surveillance, in some scenarios it might be sufficient for digital rights management schemes to degrading the visual quality to an extent where a pleasant viewing experience is no longer possible ("sufficient encryption"). Only transparent encryption guarantees a minimum quality of the preview image (the encrypted image transparently decoded).

We can summarize the following distinct application scenarios and requirements as follows:

- Highest Level Security/Cryptographic Security
  Applications that require a very high level of security, no information about the plaintext (image and compressed file) shall be deducible from the ciphertext.
- Content Security/Confidentiality
  Information of the plaintext may leak, but the image content must not be discernible.
- Sufficient encryption/Commercial application of encryption
  The content must not be consumable due to the high distortion (DRM systems).
- Transparent/Perceptual encryption
  A preview image has to be decodable, but the high quality version has to be hidden. Another application is privacy protection.

2.2 Selective/partial and lightweight encryption

In order to serve the purpose of reducing computational effort in the encryption process, more efficient methods as opposed to full encryption with cryptographically strong ciphers have been designed. Such systems—often denoted as "selective/partial" or "soft" encryption systems—usually trade off security for runtime performance, and are therefore—in terms of security—somewhat weaker than the naive method. Whereas selective or partial encryption approaches restrict the encryption process (employing classical ciphers like AES) to certain parts of the visual data by exploiting application-specific data structures or by encrypting only perceptually relevant information (e.g., encryption of I-macroblocks in MPEG, packet data of leading layers in JPEG2000), the soft encryption approach employs weaker encryption systems (like permutations) to accelerate the processing speed. Often, selective/partial encryption or soft encryption are termed "lightweight encryption".

2.3 Bitstream-oriented versus compression-integrated encryption

Bitstream-oriented techniques only operate on the final compressed stream, i.e., the codestream. Although they may parse the codestream and for example use meta-information from the codestream, they do not access the encoding (or decoding) pipeline. Classical methods for encryption fall into this category, and also many selective/partial encryption schemes that only encrypt parts of the codestream.

Compression-integrated techniques apply encryption as part of the compression step, sometimes going so far that part of the compression actually is the encryption. One possibility is to apply classical encryption after the transform step (which in most cases inevitably destroys compression performance). For other approaches the transform step is also the encryption step at the same time. Another possibility to achieve compression-integrated encryption is by selecting the transform domain to be used for encoding based on a key.

2.4 On-line/off-line scenario

Two application scenarios exist for the employment of encryption technology in multimedia environments [46] if we distinguish whether the data is given as plain image data (i.e., not compressed) or in form of a codestream resulting from prior compression. In applications where the data is acquired before being further processed, the plain image data may be accessed directly for encryption after being captured by a digitizer. We denote such applications as "on-line". Examples for this scenario are video conferencing and on-line surveillance. On the other hand, as soon as visual data has been stored or transmitted once, it has usually been compressed in some way. Applications where codestreams are handled or encrypted are denoted "off-line". Examples are video on demand and retrieval of medical images from a database.

Note that while this distinction is related to the distinction between bitstream-oriented and compression-integrated encryption, it is a distinction by application, not by procedure. In principle, both bitstream-oriented and compression-integrated methods may be suited for either of the two scenarios. However, the application of compression-integrated methods in an off-line scenario will in general not be very efficient, for obvious reasons.

2.5 Format-compliance

The aim of format-compliant encryption is to preserve—carefully—selected parts of the (meta-)information in the codestream so that the encrypted data is compliant to the format of the unencrypted data. If format compliance is desired, the classical cryptographic approach (the naive method) cannot be employed as no (meta-)information is preserved. In many cases, header information is left in plaintext and the actual visual information is encrypted avoiding the emulation of marker and header sequences in the ciphertext parts. In this manner, the properties of the original codestream carry over to the encrypted stream. For example, rate adaptation may be done in the encrypted domain easily, provided the original codestream facilitates this functionality as well (which is true for scalable or embedded codestreams, for example). While the headers are not encrypted in most approaches proposed to date, they may be encrypted in a format-compliant way as well.

The requirement of format compliance can safely be assumed to be of great importance. Format-compliance enables the transparent application of encryption, leading to numerous benefits such as signal processing in the encrypted domain, rate adaptation, or reduction of deployment costs.

3 The JPEG2000 standard suite

JPEG2000 has 13 parts (part 7 has been abandoned). For the focus of this survey our interest is in Part 1 (the core coding system), Part 2 (extensions), Part 4 (conformance testing) and Part 8 (JPSEC).

3.1 Part 1: the core coding system

JPEG2000 [56] employs a wavelet transform; Part 1 of the standard specifies an irreversible 9/7 and a reversible integer 5/3 wavelet transform and requires the application of classical pyramidal wavelet decomposition. The components of the image (after an optional multi-component transform) are subdivided into tiles, each of these tiles is independently wavelet-transformed. For a detailed description of the data partitioning refer to [56, p. 449] or to [34, p. 42]. After the wavelet transform the coefficients are quantized and encoded using the EBCOT scheme, which renders quality scalability possible. Thereby the coefficients are grouped into codeblocks and these are encoded bitplane by bitplane, each with three coding passes (except the first bitplane). The coding passes may contribute to a certain quality layer. A packet body contains CCPs (codeblock contribution to packet) of codeblocks of a certain resolution, quality layer and precinct (a spatial inter-subband partitioning structure that contains one to several codeblocks) of a tile of a certain component. A CCP may consist of a single or multiple codeword segments. Multiple codeword segments arise when a coding pass (in the CCP) is terminated. This will happen if all coding passes are terminated (JJ2000 option: `-Cterm all`).

The JPEG2000 codestream—the standard's term for the JPEG2000 stream (cf. Sect. 3.3)—consists of headers (main header, tile headers, tile part headers) and packets that consist of packet headers and packet bodies (cf. Fig. 1). The compressed coefficient data is contained in the packet bodies.

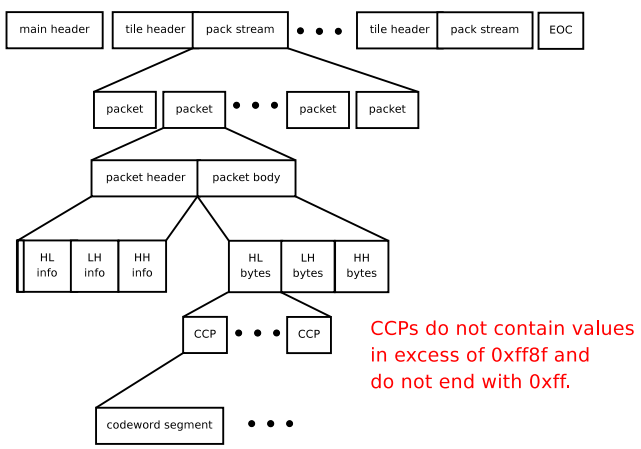


Fig. 1 Restrictions within the CCPs

The CCPs must not contain any two byte sequence in excess of `0xff8f` nor end with a `0xff` byte (bitstream compliance) [34, p. 56] . The arithmetic coding of the bitplanes is referred to as tier 1 encoding, while the partitioning of the coding passes into quality layers and the generation of the packet headers is referred to as tier 2 encoding.

3.1.1 JPEG2000 headers

The main header and tile-part header contain information about the specific compression parameters (e.g., image size, tile size, number of components, codeblock size, wavelet filters, ...). The packet header contains the following data items: inclusion information for each codeblock (does the codeblock contribute to this packet?), the lengths of the CCPs, the number of contributed coding passes for each codeblock, and the number of leading zero bitplanes for each codeblock (LZB).

3.2 Part 2: extensions

Part 2 of JPEG2000 specifies extended decoding processes, an extended codestream syntax containing information for interpreting the compressed image data, an extended file format, a container to store image meta-data and a standard set of image meta-data. The extensions of Part 2 allow employing custom wavelet transforms and arbitrary decomposition structures.

The extended coding processes are beneficial for certain applications, such as fingerprint compression and medical image compression [4,40,57]. As fingerprints and medical images contain sensitive information, security concerns naturally arise.

3.3 Part 1 and 4: bitstream, format and JPEG2000 compliance

The term "bitstream" in its common meaning refers to an arbitrary stream of bits. In the MPEG-4 standards bitstreams denote the compressed video stream [33]. The term "bitstream-oriented encryption" refers to the encryption of the compressed stream, i.e., the JPEG2000 codestream. However, in the JPEG2000 standard the term "bitstream" has a precisely defined alternate meaning. According to the JPEG2000 standard [30], "bitstream" is defined in the following way: "The actual sequence of bits resulting from the coding of a sequence of symbols. It does not include the markers or marker segments in the main and tile-part headers or the EOC marker. It does include any packet headers and in stream markers and marker segments not found within the main or tile-part headers" [30, p. 2].

Sequences in excess of `0xff8f` are used to signal in-bitstream markers and marker segments and therefore must not be generated in the encryption process (schemes fulfilling this requirement and avoiding `0xff` bytes at the end of an encryption unit, i.e., codeword segment, CCP, or packet body, and preserving the length of the encryption unit are denoted bitstream-compliant). An encryption scheme delivering a valid JPEG2000 codestream (in the sense that it is decodable by the reference software) is denoted as format-compliant. Part 4 of the JPEG2000 standard suite (conformance testing) [31] defines the term "compliance" for JPEG2000 decoders and encoders. While JPEG2000 decoders have to decode certain test sets within given error bounds in order to be compliant, the only requirement for encoder compliance is to produce compliant codestreams (decodable by the reference software); any other requirements using quality criteria are not part of the standard [31, p. 30]). JPEG2000 compression with a compliant encoder, which is followed by encryption that results in a decodable JPEG2000 codestream, is therefore JPEG2000 compliant in the sense of [31].

3.4 Part 8: JPSEC

JPEG2000 Part 8 (JPSEC) has only recently become an official ISO standard (ISO/IEC 15444-8 [32]). The standardization process started with a call for proposals in March 2003 and since then quite a number of contributions have been made [1,2,5,12,13,62,63]. JPSEC is an open security framework for JPEG2000 that offers solutions for

- Encryption
- Conditional access
- Secure scalable streaming
- Verification of data integrity
- Authentication

Encryption, conditional access and secure scalable streaming overlap with the topic of this survey.

3.4.1 JPSEC architecture

The JPSEC framework offers a syntax for the definition of JPEG2000 security services. This syntax specifies the JPSEC codestream. A JPSEC codestream is created from either an image, a JPEG2000 codestream or an existing JPSEC codestream. The last case applies if several security tools are applied subsequently.

Currently security tools are grouped into three types of tools, namely template, registration authority, and user-defined tools. Template tools are defined by the normative part of the standard, registration authority tools are registered with and defined by a JPSEC registration authority, and user-defined tools can be freely defined by users or applications. The standard defines a normative process for the registration

of registration authority tools. The registration authority and the user-defined tools enable the application of custom and proprietary encryption methods, which leads to a flexible framework.

In the following section a more detailed summary of the JPSEC codestream syntax and semantics is given.

3.4.2 The JPSEC syntax and semantics

JPSEC defines a new marker segment for the JPEG2000 main header (SEC marker segment), which is preceded only by the SIZ marker segment [32, p. 9]. Therefore the information of the SIZ marker segment is always preserved by JPSEC encryption. The SIZ marker contains information about the number of components of the source image, their resolutions (subsampling factors), their precision, as well as the chosen tile size. Note that this information is always accessible even if the most secure settings are chosen for JPSEC encryption (e.g., AES encryption of the entire remaining codestream).

The first SEC marker segment in a JPSEC codestream defines if INSEC marker segments are employed, the number of applied tools and the TRLCP format specification (the number of necessary bits to specify tile, resolution, layer, component, and precinct uniquely; in conjunction these indices uniquely identify a packet). The INSEC marker segment is used in conjunction with a non-normative tool and it may be present in the bitstream. The INSEC marker segment makes use of the fact that the JPEG2000 decoder stops decoding if a termination marker (a sequence in excess of `0xff8f`) is encountered. Thus encryption specific information can be placed directly in the JPEG2000 bitstream. The application of INSEC markers, though not without merits, also leads to certain drawbacks. First, the preservation of JPEG2000 format compliance, as defined in Sect. 3.3, requires the packet header to be changed (cf. to the approach of [25] in Sect. 5.2 for details). Second, if no specifically tailored encryption routines are employed (bitstream-compliant), the INSEC marker segment may not be parsed correctly. Therefore a useful application of INSEC markers is together with bitstream-compliant encryption algorithms (see Sect. 5.3).

The SEC marker segment also contains a list of tool specifications (one for each tool). The JPSEC tool specification follows a normative syntax and defines which type of tool is applied (either normative or non-normative), which specific tool is used, where it is applied (ZOI:= zone of influence) and its parameters (e.g., keys, initialization vectors, …).

The ZOI can be specified via image or non-image related parameters. A ZOI specification consists of one or multiple zone descriptions, the ZOI is the union of all the zones. Each zone is described by several parameters of a description class (image related or non-image related). For image related parameters a zone is the region where all the parameters are met (intersection). If multiple non-image related parameters are given, the specified regions should correspond to each other in a one to one manner, e.g., if packets and byte ranges are employed, the byte ranges specify the packet borders. In this manner the ZOI can be used to store meta-data of the codestream, e.g., where certain parts of the image are located in the codestream.

The image related description class allows to specify a zone via image regions, tiles, resolution levels, layers, components, precincts, TRLCP tags, packets, subbands, codeblocks, ROIs (regions of interest), and bitrates. The non-image related description class allows to specify packets, byte ranges (padded and unpadded ranges if padding bytes are added), TRLCP tags, distortion values, and relative importance. The distortion value and the relative importance may be set to signal to a decoder or adaptation element the importance of the specified ZOI. While the distortion value gives the total squared error if the corresponding ZOI is not available for decoding, the relative importance field is not tied to a specific quality metric. By employing these fields efficiently and in an informed way, transcoding can be conducted even if the JPSEC codestream consists of fully encrypted segments (see Sect. 3.4.3). The parameters of a tool also have to be specified; for normative tools the parameter description follows a distinct syntax, while non-normative tools may define their own syntax and semantics.

The parameter description for JPSEC normative tools consists of a template identifier and the corresponding template parameters for the tool, the processing domain, the granularity, and a list of the actual parameter values VL (initialization vectors, MAC values, digital signatures, …).

There are three basic templates for JPSEC normative tools, namely the decryption template, the authentication template and the hash template. These are further subdivided. For the decryption template a block cipher template, a stream cipher template, and an asymmetric cipher template are defined. Several block ciphers are available (AES, TDEA, MISTY1, Camellia, Cast-128, and Seed), one stream cipher (SNOW 2), and one asymmetric cipher (RSA-OAEP).

The processing domain is used to indicate in which domain the JPSEC tool is applied. The possible domains are: pixel domain, wavelet domain, quantized wavelet domain, and codestream domain.

The granularity defines the processing order (independently of the actual progression order of the JPEG2000 codestream) and the granularity level. The granularity level may be component, resolution, layer, precinct, packet, subband, codeblock, or the entire ZOI. Thus the ZOI specifies a subset of the image data (either in the image domain or in the compressed domain), while the processing order specifies in which order these data are processed (which may differ from the progression order of the protected JPEG2000 codestream). The granularity level specifies the units in which the data are processed (which can be a further subset of

the data specified through the ZOI). The list of parameter values VL contains the appropriate parameter for each of these processing units.

The following example is given in the standard [32] to illustrate the relationship between ZOI, processing order, granularity level and the list of parameter values: A JPEG2000 codestream has been encoded with resolution progression (RLCP) and 3 resolution levels and 3 layers. The ZOI is defined by resolutions 0 and 1. The processing order is layer and the granularity level is resolution. Figure 2 illustrates the process, the value list VL would contain hash values (if hashing is applied).

The granularity syntax is employed by secure scalable streaming (SSS) as proposed in [1,2,62,63]. Its implementation within JPSEC is discussed in the next section.

3.4.3 JPSEC and bitstream-oriented encryption

The normative tools of JPSEC enable the rearrangement of JPEG2000 data (except the main header) into segments. It is possible to conduct the rearrangement across packet borders. The segments are then encrypted (see Fig. 3). Segment-based encryption enables very efficient secure transcoding, i.e., SSS, because the meta-data of segmentation and encryption is stored in the SEC marker segment. Hence a JPSEC transcoder only needs to parse the main header for the SEC segment and truncate the JPSEC codestream at the according position. Compression performance is hardly influenced by this approach.

The rest of the JPEG2000 codestream (tile headers, packet headers and packet bodies) is reassembled into segments. The advantage of this approach is a low transcoding complexity, while a disadvantage is that rate adaption can only be done by a JPSEC-capable transcoder (but not by a transcoder that is only JPEG2000-compliant). In general, the advantages of format-compliant encryption are lost, but scalability is preserved to a definable level.

Format-compliant bitstream-oriented encryption schemes (see Sect. 5) can be implemented as non-normative tools.

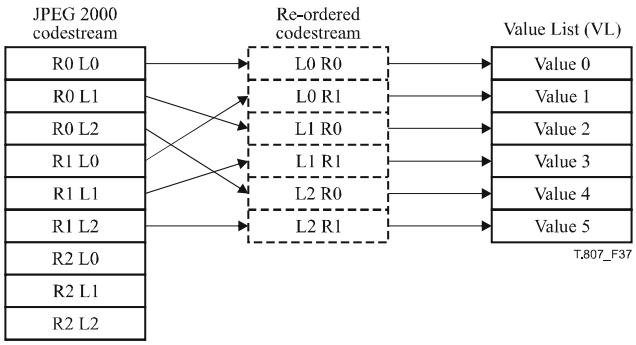


Fig. 2 Granularity level is resolution

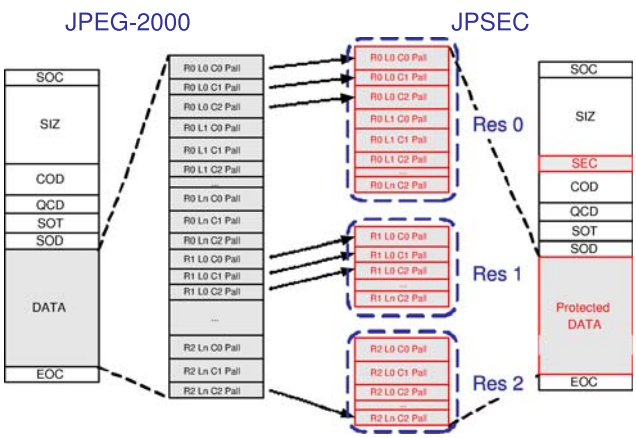


Fig. 3 Segment-based encryption

3.4.4 JPSEC and compression-integrated encryption

JPSEC allows to specify the ZOI via image related parameters. The area specified by the ZOI may be encrypted by employing a normative tool. Normative tools allow the specification of a processing domain, e.g., pixel domain, wavelet domain, quantized wavelet domain, or codestream domain. If the wavelet domain or quantized wavelet domain is chosen, the processing domain field indicates whether the protection method is applied on the sign bit or on the most significant bit [32, p. 35]. Hence the encryption of rather freely definable portions of the wavelet transformed data is possible.

3.5 The interplay of JPEG2000, JPSEC and JPEG2000 encryption

JPSEC can be used to secure JPEG2000 codestreams. Annex C of the JPSEC standard [32, p. 91] elaborates in more detail on the interoperability of JPSEC and the other parts of the JPEG2000 standard suite. As a JPEG2000 Part 1 decoder will skip marker segments that it does not recognize (see [34, p. 28]), it is possible to place the SEC marker segment in the main header of JPEG2000 codestream and still preserve compliance to JPEG2000 Part 1. The term "Part 1 compliance" is defined in [32, p. 91] for JPSEC codestreams that have a strictly defined behavior for a JPEG2000 Part 1 decoder. Note that this definition of "Part 1 compliance" is stricter than the definition of "format compliance" for JPEG2000 given in Sect. 3.3. However, the definition given in Sect. 3.3 is sufficient for assessing the compliance for JPEG2000 encoders according to Part 4 of the standard and will therefore be sufficient for assessing format compliance. Many of the JPEG2000 encryption approaches will produce codestreams that are in accordance with both compliance definitions, e.g., the encryption via a random permutation of the wavelet coefficients (see Sect. 6.2.2).

In summary, JPSEC can be used to format-compliantly signal all the necessary parameters of a format-compliant encryption scheme.

The extended coding system of JPEG2000 Part 2 offers vast parameter spaces, and thus keeping the actual parameters secret (the chosen decomposition structure, or the wavelet filter) can be employed as a form of compression-integrated encryption. The advantage of this approach is that no additional decompression/decryption software is necessary, only the parameters have to be encrypted and decrypted.

Although the JPSEC standard has been tailored to JPEG2000 Part 1 codestreams, it is reasonable to employ the JPSEC syntax for the encryption of JPEG2000 Part 2 codestreams (e.g., if secret JPEG2000 Part 2 compression options are employed, the corresponding byte ranges containing these parameters are encrypted). Annex C.2 of the JPSEC standard discusses the interoperability with Part 2 and mentions that the usage of JPSEC can be signalled via Part 2 (CAP marker segment).

3.6 Application of the JPEG2000 standard

JPEG2000 does not yet dominate the mass market, but there are several areas where it has been widely adopted. Most interesting for the scope of this survey is the Digital Cinema Specification that defines JPEG2000 as intra-frame codec. As content and copyright protection play a major role in this area there is extensive coverage of security issues in the DCI specification.

3.6.1 DCI's digital cinema specification

Despite the extensive coverage of security issues in [8], the defined encryption methods are conventional. The digital video is divided into reels of 10–20 min. These reels consist of several track files that may contain image, audio, subtitle and other meta-data [8, p. 44]. A track file starts with a file header and ends with a file footer. The track file body consists of several KLV (key length value) units. The key is an identifier for the content of the KLV unit, length specifies the length of the value. For an image track file, the image data is wrapped using KLV on an image frame boundary [8, p. 47]. For encryption, KLV units are simply mapped to new K*L*V*. While K* and L* are the new identifier and length, V* is composed of cryptographic options, K, L and the encrypted V. In other words: the video is encrypted frame per frame. The application of the AES cipher in CBC mode with 128 bit keys is required.

The JPEG2000 file is fully encrypted, only its length and the fact that it is an image track file are known. Frame dropping can easily be implemented by ignoring the corresponding KLV unit. To transcode an encrypted image, its entire data has to be decrypted. Ciphertext bit errors affect one block (16 bytes) and one bit of the JPEG2000 file [51]. The KLV, CBC and JPEG2000 systems are prone to synchronization errors, e.g., bit loss. For images this method corresponds to the naive encryption approach, while for videos it is notable that the compressed frame sizes are preserved in the encrypted domain and can potentially be used as a fingerprint.

3.6.2 Software implementations

JPEG2000 Part 1 is implemented in the reference software: There is a C implementation (JasPer) and a Java implementation (JJ2000). Apart from the reference software there are several commercial implementations, e.g., Kakadu. For our experiments we employ JasPer (Version 1.900.1), JJ2000 (Versions 4.1 and 5.1), and Kakadu (Version 6.0).

4 Evaluation criteria for JPEG2000 encryption

In addition to the different categories discussed in Sect. 2, which relate to intended level of security, field of application and mode of operation, criteria for the evaluation of the different encryption schemes are necessary. While their diversity makes it hard to directly compare all schemes, there are some criteria common to all encryption schemes that can be used in an evaluative comparison.

4.1 Compression

Compression performance may suffer if encryption is applied. While most of the bitstream-oriented encryption schemes have no or only a negligible influence on the compression performance, many compression-integrated schemes may dramatically decrease the compression performance, especially if inaccurate parameters are chosen. However, the influence on compression performance may also depend strongly on the source image characteristics.

4.2 Security

Given the different security levels of various application scenarios (as defined above) the definition of security will vary. For high-level security every bit of information that is preserved during encryption reduces the security. However, none of the format-compliant encryption schemes discussed in this survey complies with these high standards. At least the main headers and the packet structure (packet body and header borders) are preserved in the encryption process. Thus linking a plaintext and a ciphertext is to some extent possible for all of the schemes.

For content security it has to be assessed if the image content is still discernible; the standard image quality metric

PSNR is not well suited for this task. There is a similar situation for sufficient encryption: it has to be assessed whether the image still has a commercial value. For transparent/perceptual encryption a certain image quality has to be preserved, but an attacker shall not be capable to further increase the image quality by exploiting all available information (e.g., the encrypted parts).

Hence, security for all but the high-level case may be defined by the level of resistance to increase the image quality by an attack. These attacks can exploit any of the preserved data, as well as context specific side channel information (e.g., some statistics of the source images may be known). This cryptoanalytic model for multimedia encryption has been proposed in [49] (furthermore a public low quality version is assumed here, which is not appropriate for the case of content security).

For these definitions of security the evaluation of image quality is necessary.

4.2.1 Evaluation of image quality

The peak-signal-to-noise-ratio (PSNR) is no optimal choice for assessing image quality. A state-of-the-art image quality measure is the structural similarity index (SSIM) [61] and it ranges, with increasing similarity, between 0 and 1. Mao and Wu [42] propose a measure specifically for the security evaluation of encrypted images that separates luminance and edge information into a *luminance similarity score* (LSS) and an *edge similarity score* (ESS). LSS behaves in a way very similarly to PSNR. ESS is the more interesting part and ranges, with increasing similarity, between 0 and 1. We use the weights and blocksizes proposed by [42] in combination with Sobel edge detection.

4.3 Complexity

The proposed schemes are diverse, some need to run through the entire JPEG2000 compression pipeline (compression-integrated), others do not (or only partly). For some compression-integrated proposals the complexity of the compression process is increased (as for wavelet packets as described in Sect. 6.1.1), while for others the compression complexity remains unchanged (as for parameterized filters as described in Sect. 6.1.2).

Initially, one would assume that JPEG2000 encryption has to compete against conventional encryption (naive approach) in terms of runtime performance. However, most of the (runtime) benefits of JPEG2000 specific encryption schemes are due to the preservation of image and compressed domain properties in the encrypted domain. Probably the most important feature is scalability. If scalability is preserved, rate adaptation can be conducted in the encrypted domain, whereas otherwise the entire encrypted codestream needs to be decrypted. The issue of key distribution is thereby greatly simplified, as the key does not need to be present for rate adaptation. All of the discussed JPEG2000 encryption schemes preserve the scalability to some extent and thus the direct comparison of the runtime with the naive approach is not representative for the actual runtime benefits.

In order to give an estimate of runtime performance of the various JPEG2000 encryption schemes, several time estimates are needed as reference. The bitstream-oriented schemes need to identify the relevant portions of the codestream. There are three possibilities: The first is to analyze the codestream in the same manner as a JPEG2000 decoder, basically the header and the packet headers need to be decoded. An alternative is to employ SOP and EPH marker sequences to identify the relevant portions. This method is extremely simple (parsing for two byte marker sequences) compared to relatively complex packet header decoding via several tag trees and contextual codes. The third possibility is to employ JPSEC as meta-language to identify the relevant parts at the decrypter/decoder side.

For compression-integrated schemes the runtime complexity of the compression pipeline is necessary.

The following numbers are based on a test set of 1,000 images (512 × 512, 2 bpp, single quality layer) and averages of 100 trials on an Intel(R) Core(TM)2 CPU 6700 @ 2.66 GHz. The results for header decoding have been obtained by modifying the reference software JasPer (see Sect. 3.6.2) and the results for SOP/EPH parsing have been obtained by a custom implementation. Additionally, for compression and decompression the results of the Kakadu implementation are given. Empirical results for:

- time of header decoding
  very low (370.92 fps, 23.18 MB/s)
- time for SOP/EPH parsing
  extremely low (1030.93 fps, 63.84 MB/s)
- time of compression
  high (JasPer: 12.89 fps, 0.81 MB/s, Kakadu with 2 threads: 41.19 fps, 2.57 MB/s, Kakadu with 1 thread: 25.00 fps, 1.56 MB/s)
- time of decompression
  high (JasPer: 21.45 fps, 1.34 MB/s, Kakadu with 2 threads): 60.18 fps, 3.76 MB/s, Kakadu with 1 thread: 40.23 fps, 2.51 MB/s)

Compared to compression and decompression, header decoding and SOP/EPH parsing are extremely computationally inexpensive. Therefore bitstream-oriented techniques are preferable if the visual data is already compressed. However, SOP/EPH parsing is significantly less expensive than header decoding (three times less according to our results).

5 Bitstream-oriented techniques

The basic unit of the JPEG2000 codestream is a packet, which consists of the packet header and the packet body (see Sect. 3.1). Almost all bitstream-oriented JPEG2000 encryption schemes proposed in literature target the packet bodies. Format-compliance can easily be preserved by adhering to a few syntax requirements (namely those that relate to bitstream compliance: no sequences in excess of `0xff8f` are allowed and the last byte must not equal `0xff`). Scalability is thereby preserved on a packet basis. Additionally, if the packet headers are preserved, the lengths of the plaintext parts (packet bodies) have to be preserved as well. Several bitstream-compliant encryption algorithms have been proposed and are discussed in Sect. 5.3.

Scalability at an even finer level than packets can be preserved if each CCP (or even more general, each codeword segment) is encrypted independently. If encryption modes are employed that need initialization vectors (IVs), it has to be guaranteed that the IVs can be generated at the decrypting side as well—even if allowed adaptations of the encrypted codestream have been performed during transmission. The generation of truncation and cropping invariant initialization vectors is discussed in [70]. Basically a codeblock can be uniquely identified in a JPEG2000 codestream (e.g., by specifying the component, the tile, the resolution, the subband, the precinct and the codeblock's upper left coordinates) and a codeblock contribution to a packet can be uniquely identified by the quality layer. In [70] tiles are identified by their position relative to a reference grid (which is truncation/cropping invariant) and codeword segments are identified by the first contributing coding pass.

Several contributions discuss how to enable scalable access control (e.g., a user only has access to the lowest resolution, as a preview) [32, p. 65], [27,28,67] with only one single master key. This is in general achieved via hash chains and hash trees.

In the following sections, we first discuss replacement attacks and their simulation by JPEG2000 error concealment in Sect. 5.1. In Sect. 5.2 we discuss format-compliant packet body encryption algorithms which require the packet header to be modified. Note that these schemes can also be applied on a CCP or codeword segment basis, preserving scalabili-ty on a finer granularity, but requiring every CCP or codeword segment length in the packet header to be changed. Then in Sect. 5.3 packet body encryption with bitstream-compliant algorithms is discussed which allows to preserve the original packet header (again these algorithms may also be applied on a CCP or codeword segment basis). Both of these schemes preserve practically all of the packet header information, which leads to serious security problems concerning content security/confidentiality. Therefore a format-compliant packet header encryption algorithm is discussed in Sect. 5.4.

5.1 Security issues and attacks

If only parts of the JPEG2000 codestream are encrypted, these might be identified and replaced [45], thereby tremendously increasing the image quality of a reconstruction as compared to a reconstruction with the encrypted parts in place. A way to mimic these kinds of attacks is to exploit the JPEG2000 built-in error resilience tools [45]: while error resilience options are optional, they represent the same outcome that a possible attacker is likely to obtain by identifying the encrypted portions of the wavelet coefficients by means of a statistical analysis.

In the case of the JJ2000 decoder—preliminarily the error-correcting symbols have to be invoked by passing the `-Cseg_symbol on` option to the JJ2000 encoder—the erroneous bitplane and all successive bitplanes are discarded. This error concealment method protects each cleanup pass with 4 bits, as at the end of each cleanup pass an additional symbol is coded in uniform context (`0xa`). Additionally, further JPEG2000 error concealment strategies can be employed, such as predictive termination of all coding passes (invoked with `Cterminate all` and `Cterm_type predict`). Predictive termination of a coding pass protects the data with 3.5 bits in average, as error concealment information is written on the spare least significant bits of the coding pass.

It has to be noted that the JJ2000 library has minor bugs in the error concealment code (for details please cf. to [54] and [53]). The bug-fixed JJ2000 source code is available at http://www.wavelab.at/~sources/.

Attacks which use the error-concealment mechanisms to identify and replace the encrypted portions of the codestream are called error-concealment attacks or replacement attacks.

5.2 Packet body encryption with packet header modification

One of the first contributions to JPEG2000 security was made by Grosbois, et al. [25]. The packet body data is conventionally encrypted (they propose to XOR the packet body bytes with key bytes derived from a PRNG), which introduces the problem of superfluous marker generation (conventional encryption does not preserve bitstream compliance), however, this topic is not further discussed in [25]. They propose storing security information (e.g., encryption key, hash value) at the end of a codeblock's bitstream after an explicit termination marker. This method was later adapted in several contributions, namely by Dufaux and Ebrahimi [13] and Norcen and Uhl [45].

The application of this method is not as straightforward as it seems to be. "The codeword segment is considered to

be exhausted if all $L_{\max}$ bytes (all the bytes contributing to a codeword segment) are read or if any marker code in the range of $\texttt{ff90}_h$ through $\texttt{ffff}_h$ is encountered." [56, p. 483]. In practice this means that it is not sufficient to simply add explicit termination markers at the end of the codeblock's bitstream in order to add data to the codestream, furthermore the overall length of the packet has to be adjusted in the packet header. Nevertheless, it is possible to overwrite packet body data (then the packet header does not need to be changed), but this causes noise in the reconstructed image. Only if the termination marker is placed at the end of the codestream (where the desired image quality has already been reached) the image quality is not lowered. However, it has to be taken into account that the last packets will be the first to be removed in the process of rate adaptation.

The approach can avoid special encryption schemes by storing the information about superfluous markers after the explicit termination marker. Neither in [25] nor in [45] is this topic discussed further. Norcen and Uhl [45] define the encryption process, namely AES in CFB mode.

We propose a simple method to avoid marker sequences: We use the `0xff8f` sequence to signal that a sequence in excess or equal to `0xff8f` has been produced. Hence for every generated sequence in excess of `0xff8e` an additional byte has to be stored. This byte can easily be appended to the packet body, the subtraction of one (all appended bytes are in excess of `0x8e`) removes the possibility of marker code generation in the appended bytes.

In [66], Mao and Wu discuss several general communication-friendly encryption schemes for multimedia. Their work includes syntax-aware bitstream encryption with bit stuffing that can be applied to JPEG2000 as well. In the basic approach for JPEG2000, conventional encryption is applied and for every byte in excess of `0xff` an additional zero bit is stuffed in. In this way, bitstream compliance of the packet bodies is achieved. The decoder reverts this process by deleting every stuffed zero bit after a `0xff` byte in the ciphertext and then conventionally decrypting the resulting modified ciphertext.

It has to be considered that the JPEG2000 packet body has to be byte-aligned. Therefore, if the number of stuffed bits is not divisible by eight, additional bits have to be added. We propose simply filling up the remaining part with zero bits. Thereby no marker can be generated. The resulting encrypted packet body length then has to be updated in the packet header. To reconstruct the ciphertext, which is then decrypted, the bit stuffing procedure is reversed and the superfluous zero bits (outside the byte boundary) are ignored.

Compression Compression performance is negligibly reduced, depending on the method to achieve bitstream compliance. If bitstream compliance is achieved by signaling violations with `0xff8f`, one additional byte for approximately every 579 bytes is generated (on average every 256th byte is a `0xff` byte and the next byte is in 113 of 256 cases in excess of `0x8e`). If bitstream compliance is achieved via bit stuffing on average one bit every 256 bytes plus the 0–7 padding bits are added.

Security The headers are not encrypted and only slightly modified.

Performance These schemes perform very well, however, as the packet headers need to be altered, JPEG2000's tier2 decoding and encoding has to be (at least partly) conducted.

5.3 Packet body encryption with bitstream-compliant algorithms

In the following bitstream-compliant encryption algorithms are presented. Bitstream-compliant encryption algorithms differ only in terms of information leakage (amount of preserved plaintext) and computational complexity. The common properties and an experimental performance analysis of the discussed schemes are discussed in Sect. 5.3.8.

5.3.1 Conan et al. and Kiya et al.

The algorithm by Conan is not capable of encrypting all of the packet body data but can be implemented rather efficiently [6]. Only the 4 LSBits (least significant bits) of a byte are encrypted, and only if the byte's value is below `0xf0`. In this way no sequence in excess of `0xff8f` is produced. It is easy to see that no bytes are encrypted to `0xff`, because only the lower half of the bytes below `0xf0` are encrypted. The byte `0xff` is preserved. Hence a byte sequence in excess of `0xff8f` could only be produced after a preserved `0xff` byte. However, due to the bitstream-compliance the plaintext byte after a `0xff` byte is not in excess of `8f`.

Kiya et al. [35] extend this approach to an even more lightweight and flexible scheme. They propose encrypting only one randomly chosen byte out of m bytes. In this way the choice of the parameter m can trade-off security for performance. Additionally they propose a random shift of the 4 LSBits instead of encryption. The shifting operation can be applied to the 4 LSBits of all bytes, while preserving bitstream compliance.

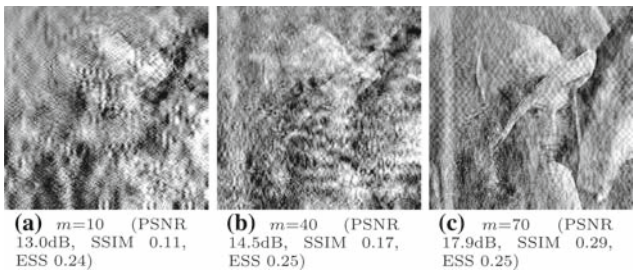
Security The information leakage is very high, even with the most secure settings more than half of the compressed coefficient data remains unencrypted. Assuming state-of-the-art underlying encryption techniques, the encrypted half bytes are irrecoverable by cryptographic means and therefore the whole arithmetic codeword is in general irrecoverable.

The shifting algorithm is less secure since there are only 4 possible ciphertexts for a half byte as compared to 16 for the encryption algorithm (which of course have to be different for every byte). Figures 4 and 5 show the result of the direct reconstruction, i.e., the full reconstruction of the encrypted codestream without trying to conceal the encrypted parts, and the concealment attack and different values of the parameter m. The shifting algorithm preserves more image plaintext than the encryption algorithm. Apart from error concealment options (segmentation symbol, predictive termination of all coding passes and SOP and EPH marker), the compression parameters have been set to JJ2000 default values, i.e., layer progression, 32 quality layers and no limit on bitrate (default bitrate is 100 bpp). If the 4 LSBits are encrypted, hardly any image information is visible for m up to 10. If the 4 LSBits are shifted, image information starts to become visible for m greater than one.

Performance Half byte encryption and permutation cannot be implemented much more efficiently than byte encryption on standard CPUs. For every encrypted byte one condition (is the byte below `0xf0`) has to be evaluated.

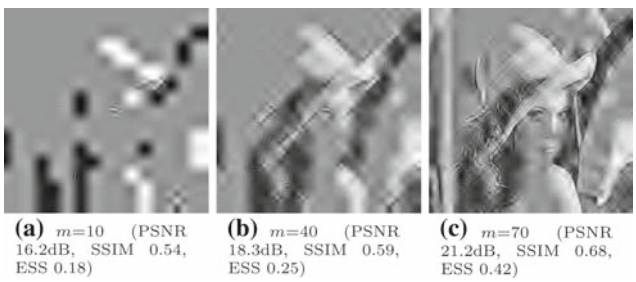
5.3.2 Wu and Ma

Wu and Ma [65] greatly reduce the amount of information leakage compared to the algorithm of Conan and Kiya (see Sect. 5.3.1). They propose two algorithms for format-compliant packet body encryption. Both algorithms only preserve the `0xff` byte and its consecutive byte and can be implemented efficiently.



(a) m=10 (PSNR 13.0dB, SSIM 0.11, ESS 0.24) **(b)** m=40 (PSNR 14.5dB, SSIM 0.17, ESS 0.25) **(c)** m=70 (PSNR 17.9dB, SSIM 0.29, ESS 0.25)

Fig. 4 Kiya: Encryption of the 4 LSBits: direct reconstruction



(a) m=10 (PSNR 16.2dB, SSIM 0.54, ESS 0.18) **(b)** m=40 (PSNR 18.3dB, SSIM 0.59, ESS 0.25) **(c)** m=70 (PSNR 21.2dB, SSIM 0.68, ESS 0.42)

Fig. 5 Kiya: Encryption of the 4 LSBits: concealment attack

Stream Cipher Based Algorithm:

Their first algorithm is based on a stream cipher (in [65] RC4 is employed). To that end a keystream is generated. By discarding `0xff` bytes, a modified keystream S is obtained. In the following, the term s_i denotes the ith byte of the keystream, m_i denotes the ith byte of the packet body (plaintext) and c_i denotes the ith ciphertext byte.

The encryption works byte by byte on the packet body in the following way:

If m_1 equals `0xff`
then $c_1 = m_1$
else $c_1 = m_1 + s_1 \mod$ `0xff`
For $i = 2$ to length

  If (m_i equals `0xff`) or (m_{i-1} equals `0xff`)
  then $c_i = m_i$
  else $c_i = m_i + s_i \mod$ `0xff`;

Every byte that is not a `0xff` byte or the successor of a `0xff` byte is encrypted to the range [`0x00`,`0xfe`]. The decryption algorithm works similarly:

If c_1 equals `0xff`
then $m_1 = c_1$
else $m_1 = c_1 - m_1 \mod$ `0xff`
For $i = 2$ to length

  If (m_i equals `0xff`) or (m_{i-1} equals `0xff`)
  then $c_i = m_i$
  else $m_i = c_i - s_i \mod$ `0xff`

This algorithm avoids producing values in excess of `0xff8f`, since no `0xffs` are produced and all two byte sequences (`0xff`, X) are preserved.

Block Cipher Based Algorithm:

This algorithm preserves all two byte sequences (`0xff`, X) as well, and basically works as follows:

Select all bytes of the packet body that are neither equal to `0xff` nor a successor of a `0xff` byte.
Split the selected bytes into blocks and encrypt them iteratively until no `0xff` is contained in the ciphertext.
Replace the selected bytes in the original block with the encrypted bytes.

When the number of selected bytes is not a multiple of the blocksize, ciphertext stealing (as described in [51]) is proposed. This algorithm does not contain any feedback mechanisms (equal packets yield equal ciphertexts, a basis for replay attacks). In [65] AES is employed.

Security Only (0xff, X) sequences are preserved, which renders the reconstruction of the image content unfeasible.

Performance The stream cipher based algorithm (as presented) needs to evaluate two conditions for every encrypted byte and needs to perform a modulo operation for every encrypted byte. Every 0xff byte has to be discarded from the keystream as well, thus slightly more encryption operations are necessary compared to conventional encryption.

For the block cipher based algorithm the probability for a ciphertext that does not contain any 0xff byte has to be assessed. In general this probability is $(\frac{255}{256})^n$, where n is the length of the plaintext and the encryption method is assumed to be uniformly distributed. This results in a success probability of approximately 96.9% for a block of 8 bytes, which increases the encryption time about 3.18% compared to the underlying encryption routine (AES with 16 bytes or Triple-DES with 8 byte are proposed in [65]). For a block of 16 bytes the success probability is 93.9%, which corresponds to an overhead of 6.46%. Thus the overhead induced by additional encryption is modest as well. However, this algorithm has additional copy operations (the bytes have to be copied to a buffer before the iterative encryption).

5.3.3 Dufaux et al.

The encryption algorithm presented by Dufaux et al. [12] is basically an improvement of the algorithm by Wu and Ma [65] in terms of reduced information leakage. Only 0xff bytes are preserved. They propose the usage of the SHA1 PRNG with a seed of 64 bit for keystream generation; however, any other cryptographically secure method of generating an appropriate keystream can also be applied. The encryption procedure is the following:

If m_i equals 0xff
then $c_i = m_i$
If m_{i-1} equals 0xff
then $c_i = m_i + s_i \mod 0x8f+1$, and $s_i \in [0x00, 0x8f]$.
If m_{i-1} does not equal 0xff
then $c_i = m_i + s_i \mod 0xff$, and $s_i \in [0x00, 0xfe]$.

The proposed method to obtain a number in the right range is the iterative generation of random numbers until a number in the right range is produced. Decryption works analogously.

Security The information leakage is further reduced compared to the algorithm by [65], only 0xff bytes are preserved (every 128th byte, cf. Sect. 5.3.8).

Performance Slightly more keystream bytes have to be used, since the 0xff bytes have to be discarded for the keystream and after a 0xff byte, bytes with values in excess of 0x8f have to be discarded. Additionally, one condition and one modulo operation have to be evaluated for every encrypted byte.

5.3.4 A JPSEC technology example

In [32, p. 72] a method for format-compliant encryption is sketched in Annex B.5 (Technology examples: Encryption tool for JPEG2000 access control). The document, however, does not contain all necessary details to implement the method. On the contrary, in [29] it is pointed out that the proof for the reversibility of the algorithm is still missing.

The encryption process is defined in the following way:

The packet body is split into two byte sequences.
Every two byte sequence of the packet body is temporarily encrypted.
If the temporary byte sequence or its *relating code* is more than 0xff8f it is not encrypted, otherwise the temporarily encrypted code is outputted as ciphertext.

If the length of the plaintext is odd it is proposed to leave the byte in plaintext or pad an extra byte. The padding of an extra byte would require the modification of the packet header. The decryption process is similarly specified:

The packet body is split into two byte sequences.
Every two byte sequence is temporarily decrypted.
If the temporary byte sequence or its *relating code* is more than 0xff8f it is not decrypted, otherwise the temporarily decrypted code is outputted as plaintext.

It is notable that the underlying encryption routine for two byte sequences must satisfy the following property $e(p) = c = d(p)$ and thus $e(e(p)) = p$. This is met by all encryption modes that xor the plaintext with a keystream (e.g., OFB mode). The term *relating code* is not further specified. Furthermore it is possible for an encrypted packet body to end with 0xff, which might lead to problems (a marker sequence at packet borders is possibly generated).

In [15] an interpretation for the term *relating code* is given which makes the scheme reversible (a proof is given):
Let P_j denote the jth plaintext two byte sequence, I_j the jth temporarily encrypted two byte plaintext sequence, C_j the jth two byte ciphertext sequence and D_j the jth temporarily decrypted ciphertext sequence. The term $X|Y$ denotes the concatenation of the second byte of X and the first byte of Y, where X and Y are arbitrary two byte sequences. If the following conditions are met, then the ciphertext C_j is set to the temporarily encrypted sequence I_j:

E1 $I_j \leq$ 0xff8f
Necessary to obtain a bitstream-compliant two byte ciphertext sequence.
E2 $P_{j-1}|I_j \leq$ 0xff8f
Necessary to ensure bitstream compliance if the previous two byte sequence has been left in plaintext.
E3 $I_{j-1}|I_j \leq$ 0xff8f
Necessary to ensure bitstream compliance if the previous two byte sequence has been replaced by the temporarily encrypted sequence.
E4 $I_j|P_{j+1} \leq$ 0xff8f
Necessary to be able to preserve the next two byte sequence in plaintext.
E5 $I_{j-1}|P_j \leq$ 0xff8f
Necessary to detect E4 for $j-1$.

In order to decrypt the jth ciphertext the following conditions have to be met:

D1 $D_j \leq$ 0xff8f
Detection of the violation of E1 (if E1 has not been met D1 is not met and the ciphertext is the plaintext).
D2 $P_{j-1}|D_j \leq$ 0xff8f
Detection of the violation of E2.
D3 $I_{j-1}|D_j \leq$ 0xff8f
Detection of the violation of E3.
D4 $D_j|C_{j+1} \leq$ 0xff8f
Detection of the violation of E4.
D5 $I_{j-1}|C_j \leq$ 0xff8f
Detection of the violation of E5.

All conditions referencing undefined bytes (e.g., P_{-1}) are by default true. Note that in the case of an even number of packet body bytes, the last two byte sequence requires special treatment. In this case the best solution (in terms of maximum encryption percentage) is to modify $E1$ and $D1$ such that a byte with value 0xff at the end is forbidden.

Security Information leakage occurs whenever a two byte sequence of plaintext is preserved. Our experiments, which implement the algorithm specified in [15], reveal that about every 128th byte is preserved (cf. Sect. 5.3.8). However, the preserved two byte sequences are not distinguishable from the encrypted sequences (compared to the previous bitstream-compliant algorithms, that always preserve the 0xff byte). Thus the algorithm is an improvement over the previous bitstream-compliant algorithms, as, for example, the two encrypted versions (different encryption keys) preserve totally different plaintext bytes.

Performance There is a slight performance overhead, due to the additional comparisons (five conditions for every two byte sequence).

5.3.5 Wu and Deng

The iterative encryption which works on CCPs was proposed by Wu and Deng in [68]. While the iterative encryption algorithm is capable of encrypting all of the packet body data, it cannot be implemented very efficiently. Contrary to most other schemes the iterative encryption algorithm does not preserve any plaintext information (except its length). The CCPs are recursively encrypted until they are bitstream-compliant. The basic encryption algorithm is the following: For all CCPs:

ccp_{mid} = encrypt(CCP)
While (isNotBitstreamCompliant(ccp_{mid}))
  ccp_{mid} = encrypt(CCP)
Output ccp_{mid} as ciphertext.

In [68] addition modulo 256^n is proposed as encryption method for the CCPs, however, encryption with the ECB mode of a blockcipher and ciphertext stealing [51] works as well and can be expected to be more efficient (and is therefore used in our experiments, see Sect. 5.3.8). Accordingly, for decryption the ciphertext is iteratively decrypted until it is bitstream-compliant. This algorithm is fully reversible and encrypts 100% of the packet body data.

Theoretically this algorithm can easily be extended to packet bodies by iteratively encrypting the packet bodies. However, the computational complexity of this algorithm will in general prevent the application of this algorithm on a packet body basis.

Compression If this scheme is applied to CCPs there is no direct influence on the compression performance, but certain parameter settings will be required to reduce the CCP lengths (e.g., enough quality layers [52]) that may reduce the compression performance.

Security There is no information leakage for the encrypted packet bodies.

Performance A detailed performance analysis of this scheme has been conducted by Stütz and Uhl [52] who show that the complexity of this algorithm increases dramatically with the length of the plaintext. Thus this algorithm is only feasible for certain coding settings which guarantee short CCP lengths, e.g., the choice of enough quality layers is necessary [52]. No stream processing is possible, the entire plaintext/ciphertext has to be kept in memory.

5.3.6 Zhu et al.

Zhu et al. [70,72] propose to apply their bitstream-compliant scheme on a codeword segment basis.

1. The plaintext is XOR ed with a keystream.
2. In this intermediate ciphertext every byte is checked to meet the bitstream compliance. If an illegal byte (its value concatenated with the value of the next byte is in excess of `0xff8f`) is found (at index *currIdx*), this and the next (if there is one) intermediate ciphertext byte are replaced with the plaintext bytes at the same location (same indices).
   (a) Now it is checked whether this replacement results in a bitstream syntax violation of the decrypted intermediate ciphertext.
      i Therefore the previous byte of the intermediate ciphertext is decrypted and together with the decrypted plaintext byte checked for bitstream compliance (are the two bytes concatenated in excess of `0xff8f`).
      ii If this two byte sequence is illegal, this byte is also replaced with the plaintext byte in the intermediate ciphertext.
      iii This procedure is conducted backwards until no more illegal byte sequences are found or a certain index ($lastModIdx+1$, which is initialized with -1) is reached.
   (b) If any byte has been replaced in (a), the intermediate ciphertext itself is checked for bitstream compliance.
      (i) Therefore the previous byte of the last intermediate ciphertext byte that has been changed to the plaintext is checked for bitstream compliance.
      (ii) If it is illegal, it is replaced with the corresponding plaintext byte.
      (iii) This procedure is conducted backwards until no more illegal byte sequences are found or a certain index ($lastModIdx + 1$) is reached.
3. (a) and (b) are repeatedly executed until no illegal bytes are found in (a) and (b).
4. The index *lastModIdx* is then set to *currIdx* and the forward search for illegal bytes in the intermediate ciphertext is continued.
5. At the end the intermediate ciphertext is outputted as ciphertext.

Why is this scheme reversible and why is the decryption algorithm the same as the encryption algorithm? If no replacements have been conducted this is obviously the case as M XOR S XOR S equals M. A precise argument for the general case may be rather complex, but the scheme relies on the simple fact that a certain plaintext sequence and a certain key sequence result in an illegal ciphertext, which may be cause for illegal intermediate decrypted sequences (see (a)) or illegal intermediate ciphertexts (see (b)), which are all "switched back" to the plaintext. As for those pairs of sequences the plaintext is preserved, this property is preserved for the ciphertext and thus perfect reconstruction is possible.

Security According to the authors 0.36% of the plaintext are preserved [70].

Performance The encryption via XOR is very fast, but the searching for bitstream syntax violations is necessary. The entire plaintext/ciphertext has to be kept in memory (no stream processing).

5.3.7 Fang and Sun

The algorithm presented by Fang and Sun in [22] does not preserve any plaintext byte sequence and can be applied to CCPs and packet bodies. Nevertheless it is a computationally rather inexpensive procedure that concurrently works on three consequent plaintext bytes (the other schemes only consider two consequent plaintext bytes). The actual encryption and decryption algorithms are rather complex, therefore we will also give their pseudo code.

The first byte is encrypted depending on the second byte. If the second byte is in excess of `0x8f`, then according to the bitstream syntax, the first byte must not be encrypted to `0xff`. If $m_1 + s_1$ equals `0xff`, then c_1 is set to $m_1 + 2s_1$, which cannot yield `0xff` too (only possible if s_1 is zero and m_1 is `0xff`, but then, according to the bitstream compliance, the second byte cannot be in excess of `0x8f`).

The second byte (and all following, except the last one) is encrypted depending on the previously encrypted plaintext byte, the previous cipher byte and the encryption of the previous byte (if it employed double encryption), the current plaintext byte and the next plaintext byte. There are basically three cases:

1. If m_{i-1} or c_{i-1} are `0xff` then the current byte is encrypted to the range `0x00` to `0x8f` (this is possible because both facts indicate that the current plaintext byte is not in excess of `0x8f`).
2. If the current byte is in excess of `0x8f` and the previous byte has been encrypted twice, then this property has to be preserved (encryption in the range `0x90` to `0xff`) to signal the double encryption in the decryption process. The next plaintext byte has to be considered as well; if it is in excess of `0x8f`, then the current cipher byte must not become `0xff`, which is again avoided by double encryption.

3. In all other cases the byte is encrypted and double encryption is conducted if the next byte is in excess of 0x8f and the cipher byte would be 0xff.

The last byte is encrypted such that it is ensured that the cipher byte is in the same range (either 0x00 to 0x8f or 0x90 to 0xff).

We can confirm that this encryption process is reversible (also experimentally). The pseudo code of the encryption and the decryption algorithm is given.

The encryption algorithm works in the following way:

$c_1 = c_{mid} = (m_1 + s_1) \mod 256$
If c_{mid} equals 0xff and $m_2 \geq$0x90
then $c_1 = c_{mid} + s_1 \mod 256$
For $i = 2$ to $length - 1$

If m_{i-1} equals 0xff or c_{i-1} equals 0xff
then $c_i = (m_i + s_i) \mod 0x90$, $c_{mid} = 0x00$
else
if c_{mid} equals 0xff
then $c_{mid} = (m_i - 0x90 + s_i) \mod 0x70 + 0x90$
If c_{mid} equals 0xff and $m_{i+1} \geq 0x90$
then $c_i = (c_{mid} - 0x90 + s_i) \mod 0x70 + 0x90$
else $c_i = c_{mid}$
else $c_{mid} = (m_i + s_i) \mod 256$
If c_{mid} equals 0xff and $m_{i+1} \geq 0x90$
then $c_i = (c_{mid} + s_i) \mod 256$
else $c_i = c_{mid}$

If $m_{length} < 0x90$
then $c_{length} = (m_{length} + s_{length}) \mod 0x90$
else $c_{length} = (m_{length} - 0x90 + s_{length}) \mod 0x70 + 0x90$

If c_{length} equals 0xff
then $c_{length} = (c_{length} - 0x90 + s_{length}) \mod 0x70 + 0x90$

The decryption algorithm works in the following way:

$m_1 = m_{mid} = (c_1 - s_1) \mod 256$
If m_{mid} equals 0xff and $c_2 \geq 0x90$
then $m_1 = (m_{mid} - s_1) \mod 256$

For $i = 2$ to $length - 1$

If m_{i-1} equals 0xff or c_{i-1} equals 0xff
then $m_i = (c_i - s_i) \mod 0x90$, $m_{mid} = 0x00$
else
if m_{mid} equals0xff
then $m_{mid} = (c_i - 0x90 - s_i) \mod 0x70 + 0x90$
If m_{mid} equals 0xff and $c_{i+1} \geq 0x90$
then $m_i = (m_{mid} - 0x90 - s_i) \mod 0x70 + 0x90$
else $m_i = m_{mid}$
else $m_{mid} = (c_i - s_i) \mod 256$
If m_{mid} equals 0xff and $c_{i+1} \geq 0x90$
then $m_i = (m_{mid} - s_i) \mod 256$
else $m_i = m_{mid}$

If $c_{length} < 0x90$
then $m_{length} = (c_{length} - s_{length}) \mod 0x90$
else $m_{length} = (c_{length} - 0x90 - s_{length}) \mod 0x70 + 0x90$

If m_{length} equals 0xff
then $m_{length} = (m_{length} - 0x90 - s_n) \mod 0x70 + 0x90$

Security No byte sequence is preserved. However, from a cryptographic point of view there is a small weakness in this scheme.

The encryption operation $(c_i + s_i) \mod 0x90$ introduces a bias and therefore does not meet high cryptographic standards. The same holds for the operation $(m_{mid} - 0x90 + s_i) \mod 0x70 + 0x90$. However, this bias can be removed by requiring the proper range for s_i to be 0x00 to 0x8f in the first case and 0x00 to 0x6f in the second case. This can easily be integrated into the algorithm by simply ignoring out-of-range keystream bytes in case of an encryption to a restricted range. In case of encryption to the range 0x00 to 0x6f it is more efficient to halve the key byte before testing its range.

The answer to the question to which extent plaintext information is preserved in this scheme is beyond the scope of this survey paper. It is, however, intuitively dubious that every possible codeword has the same probability of becoming a plaintext's ciphertext. Nevertheless it needs to be pointed out that the information leakage is considered to be less than in all other algorithms except the iterative algorithm.

Performance Several conditions have to be evaluated in order to encrypt a single byte. If bias is to be prevented some keystream bytes have to be discarded.

5.3.8 Discussion of packet body encryption with bitstream-compliant algorithms

Several properties are shared by all approaches that employ bitstream-compliant algorithms.

Compression There is no influence on compression performance if bitstream-compliant encryption algorithms are applied.

Security The packet headers are preserved if bitstream-compliant encryption algorithms are applied (as proposed in

literature). There are known attacks (e.g., the error concealment attack) concerning selective/partial application of these bitstream-compliant schemes (see Sect. 5.1). The packet body encryption with bitstream-compliant encryption algorithms is not secure under IND-CPA (Indistinguishability under chosen-plaintext attack), as a potential attacker is very likely to successfully identify the corresponding ciphertext for a plaintext compressed image.

Most of the proposed schemes preserve plaintext bytes or properties, which is not a major concern as the headers and the packet headers already deliver a distinct fingerprint of the JPEG2000 codestream [15] (which is preserved in any case for all of the schemes, cf. Sect. 5.4).

However, only the iterative encryption algorithm of Wu and Deng is expected to be secure against IND-CPA attacks (only considering a packet body and disregarding the fingerprint obtained by the headers and packet headers).

In the following we present an empirical evaluation of information leakage of the presented bitstream-compliant encryption algorithms.

Empiric Information Leakage: In order to assess the amount of information leakage we give the average number of bytes until one byte is preserved.

Average number of bytes for one byte preservation	
Conan and Kiya ($m = 1$)	2
Mao and Wu	125.61
Dufaux	251.22
JPSEC techn. example	128.45
Zhu	312.50

The algorithm by Conan and Kiya preserves at least half of the plaintext, hence linking ciphertext and plaintext is obviously trivial. Plaintext two byte sequences starting with `0xff` are preserved in the ciphertext for the two algorithms by Mao and Wu, while only the `0xff` bytes are preserved for the algorithm by Dufaux et al. Hence plaintext and ciphertext have the same number of these sequences or `0xff` bytes at the same positions, which greatly simplifies the linking of the two. For the JPSEC Technology Example algorithm it is not known which sequences are preserved since the decision if a byte is preserved depends on the temporarily encrypted bytes (unknown to an attacker) as well. Hence the linking of plaintext and ciphertext has to exploit higher correlation between the two, which renders the process more complicated and less certain. Zhu's algorithm significantly reduces the information leakage (but stream processing is no longer possible). The algorithm by Fang and Sun does not preserve any plaintext byte but preserves some of the properties, which can be again exploited by a statistical analysis. In detail there are 4,789 bytes encrypted to the range `0x90` to `0xff` and 17,159 bytes encrypted to the range `0x00` to `0x8f` of a total of 2,782,951 encrypted bytes. The certainty of the linking is expected to be further reduced.

Performance For all of the presented bitstream-compliant encryption algorithms, except the iterative encryption algorithm (see Sect. 5.3.5), the throughput (encrypted bytes per second) is independent of the plaintext length (disregarding the initialization overhead for the underlying encryption routine). All of the algorithms employ a cryptographic primitive (stream cipher, block cipher, secure random number generator) to obtain cryptographically secure randomness. However, the specific choice of primitives to be employed varies greatly.

In order to experimentally assess the runtime performance of the bitstream-compliant encryption algorithms discussed in this survey a single source of randomness is applied, namely AES. If a stream cipher is employed in the original contribution, AES is used in OFB mode to produce the keystream (in case of the application of a block cipher AES is used directly in ECB mode). In the following table stable results (128 MB of plaintext data have been encrypted 150 times to obtain these results) for the throughput of the bitstream encryption algorithms are presented.

Throughput of bitstream-compliant encryption	
AES OFB	42.71 MB/s
Conan and Kiya ($m = 1$)	37.87 MB/s
Conan and Kiya ($m = 10$)	156.63 MB/s
Conan and Kiya ($m = 40$)	606.83 MB/s
Wu and Ma Stream	27.60 MB/s
Wu and Ma Block	1.53 MB/s
Dufaux	28.18 MB/s
JPSEC techn. example	37.81 MB/s
Zhu	29.30 MB/s
Fang and Sun	27.95 MB/s

For a large test set of 1,000 images the following throughputs have been achieved with SOP/EPH marker parsing (including all file reads and parsing). Results are presented for 1, 20 and 100% encryption of the packet body data, thereby covering the encryption percentage for all of the feasible application scenarios (see Sect. 5.5).

1% Encrypted	
Conan and Kiya ($m = 1$)	41.13 MB/s
Mao and Wu Stream	40.69 MB/s
Mao and Wu Block	40.88 MB/s
Dufaux	40.66 MB/s
JPSEC techn. example	41.01 MB/s
Zhu	40.87 MB/s
Fang and Sun	40.30 MB/s

20% Encrypted	
Conan and Kiya ($m = 1$)	33.41 MB/s
Mao and Wu Stream	29.24 MB/s
Mao and Wu Block	16.19 MB/s
Dufaux	29.67 MB/s
JPSEC techn. example	33.48 MB/s
Zhu	31.66 MB/s
Fang and Sun	29.73 MB/s

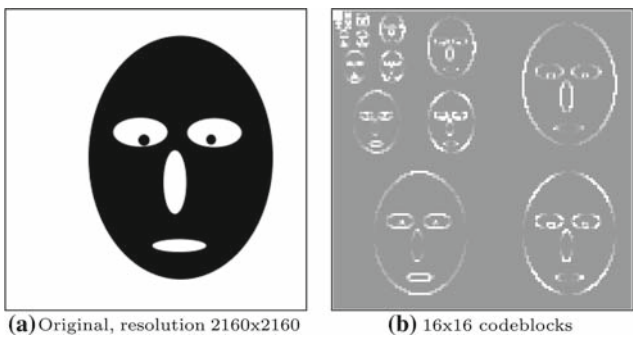
100% Encrypted	
Conan and Kiya ($m = 1$)	19.27 MB/s
Mao and Wu Stream	14.05 MB/s
Mao and Wu Block	3.83 MB/s
Dufaux	14.39 MB/s
JPSEC techn. example	19.72 MB/s
Zhu	17.24 MB/s
Fang and Sun	14.67 MB/s

The application of the iterative encryption algorithm (see Sect. 5.3.5) is not feasible in general as its complexity depends on the plaintext length. At a plaintext length of 4,000 bytes the iterative encryption algorithm achieves a throughput of only 0.07 MB/s.

Another important aspect is memory consumption. All but three algorithms (the iterative encryption algorithm by Wu and Deng, the algorithm by Zhu et al., and the block cipher based algorithm by Wu and Ma) are capable of stream processing (requiring only a few state variables), while the memory consumption of these three algorithms increases linearly with the plaintext length.

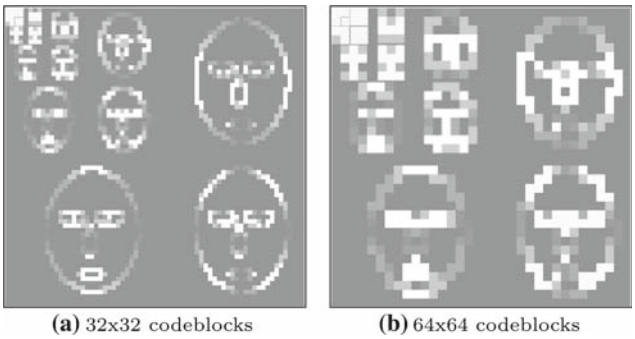
5.4 Format-compliant packet header encryption

In Sect. 3.1.1 we discussed the structure of the JPEG2000 packet headers. These contain crucial (even visual) information of the source image. Especially for high-resolution images, content security/confidentiality can not be met without encrypting the leading zero bitplane information in the packet headers (see Figs. 6, 7).



(a) Original, resolution 2160x2160 **(b)** 16x16 codeblocks

Fig. 6 The LZB information of a high resolution image



(a) 32x32 codeblocks **(b)** 64x64 codeblocks

Fig. 7 The LZB information of a high resolution image

In [15], Engel et al. propose format-compliant transformations for each piece of information contained in the packet header. These transformations make use of a random keystream, the knowledge of which allows the decoder to obtain the original packet header. The resulting codestream is format-compliant.

CCP Lengths and Number of Coding Passes: JPEG2000 explicitly signals both the number of coding passes and the length of each codeblock contribution.

The algorithm described in [15] redistributes lengths and coding passes among the codeblocks in a packet. The procedure in pseudo-code is given below. v[] is a vector of non-zero positive integers (indexing starts at 1).

```
shuffle (v)
borders = size(v) − 1
For i = 1 to borders

  sum = v[i] + v[i + 1]
  r = ⌊random(0,1)*sum⌋
  newBorder = ((v[i] + r) mod (sum-1)) + 1;
  v[i] = newBorder;
  v[i+1] := sum  −  newBorder

shuffle(v)
```

The transformation can be reversed easily by unshuffling the input, traversing it from end to start, using the random numbers in reverse order, setting newBorder as:

```
newBorder := (v[1] − r − 1) mod (sum − 1)
if (newBorder ≤ 0)
then newBorder = newBorder + (sum − 1)
```

and finally unshuffling the result again.

Leading Zero Bitplanes: The number of leading zero bitplanes (LZB) for each codeblock is coded by using tag trees [56]. As discussed above, this information is even more

critical than the other classes of header information, as by using the number of LZB an attacker can obtain information on the visual content of the encrypted image (for small codeblock sizes or high resolutions).

In [15] a random byte is added to the number of leading zero bitplanes modulo a previously determined maximum number. For decoding, the random byte is subtracted instead of added. The maximum number of skipped bitplanes needs to be signaled to the decoder, e.g., by inserting it into the key or by prior arrangement.

Inclusion Information: Each packet contains the inclusion information for a certain quality layer for all codeblocks in a precinct. There are four types of inclusions that a codeblock c can have in packet p.

The sequence of inclusion information of each codeblock is coded depending on the type of inclusion.

In [15], an algorithm is presented that allows to permute inclusion information for each packet in such a way that the original inclusion information cannot be derived without the key and that the resulting "faked" total inclusion information complies with the semantics of JPEG2000.

Combined Format-Compliant Header Transformation: The format-compliant transformation of the different pieces of information in the packet headers can be combined. The format compliance of the combined format-compliant header encryption has been verified experimentally by decoding the encrypted codestreams with the reference implementations JasPer and JJ2000.

Compression There is basically no influence on compression performance.

Security The visual information contained in LZB information is effectively encrypted (content security/confidentiality can be achieved). Even if packet body based encryption and format-compliant header encryption are combined, security under IND-CPA can still not be achieved as the packet borders are preserved.

Performance As packet header data is only a small fraction of the actual codestream, format-compliant header encryption only introduces a small overhead.

5.5 Application of bitstream-oriented encryption

Bitstream-oriented JPEG2000 encryption is capable of meeting most of the quality and security constraints of the different applications (cf. Sect. 2.1).

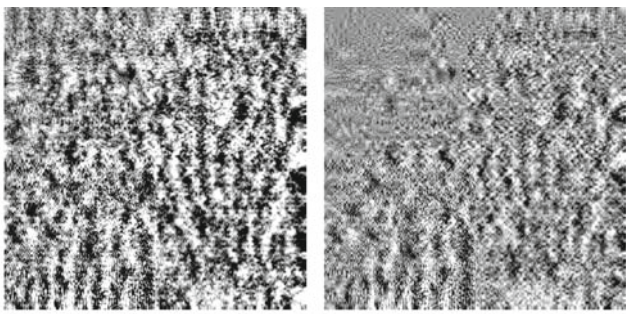
Content security/confidentiality can be achieved by encrypting all of the packet body and packet header data (this can be done format-compliantly).

Sufficient encryption can be achieved by encrypting the packet body data. Depending on the desired level of protection, partial/selective application of bitstream-oriented is feasible.

Transparent/perceptual encryption is feasible as well, via the partial/selective application of format-compliant bitstream-oriented schemes.

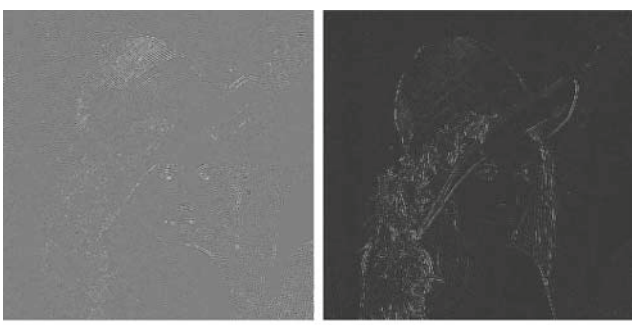
Only the highest level of security cannot be achieved, as certain properties of the image (e.g., its compressibility, truncation points, …) will always be preserved.

Norcen and Uhl evaluate JPEG2000 bitstream-oriented encryption for content security/confidentiality [45] and show that it is sufficient to encrypt the first 20% of the JPEG2000 codestream (lossy at a rate of 0.25 or lossless) in order to confidentially hide all image information. Figure 8 shows the direct reconstruction (i.e., a reconstruction with the encrypted parts in place) and the corresponding erroneous error-concealment attack (without the bug-fix mentioned in Sect. 5.1). However, if correct error concealment (i.e., a successful attack) is applied, it turns out that the rule of thumb does not hold anymore, as illustrated in Fig. 9 for both layer and resolution progression. We observe that the SSIM index is capable of measuring the similarity even for very low quality images in contrast to the PSNR and the ESS (see Figs. 8, 9). These results also relativize the claim of data



(a) Encryption: PSNR 8.4db, SSIM 0.03, ESS 0.23 **(b)** Erroneous attack: PSNR 9.7db, SSIM 0.05, ESS 0.23

Fig. 8 Confidentiality with JPEG2000: 20% encrypted



(a) Layer prog. PSNR 14.5db, SSIM 0.57, ESS 0.0 **(b)** Res. prog.: PSNR 14.5db, SSIM 0.62, ESS 0.0

Fig. 9 Concealment attack: 20% encrypted, 2bpp

confidentiality for the technology example of Annex B.10 [32, p. 85] (in this example only 1% of JPEG2000 data is encrypted). In [54] Stütz et al. give a more detailed examination of this topic, where they conclude that partial/selective encryption of JPEG2000 cannot guarantee confidentiality.

Transparent/perceptual encryption via bitstream-oriented JPEG2000 encryption has been evaluated by Obermaier and Uhl [59]. The packet body data is encrypted starting from a certain position in the codestream up to the end. This procedure allows the reconstruction of a low quality version from the encrypted codestream. In [59] the impact of the choice for the start of encryption is evaluated for different progression orders, namely resolution and layer progression. A drawback of their approach is that most (more than 90%) of the JPEG2000 codestream has to be encrypted.

More efficient solutions both in terms of computational complexity and reduced deployment cost have been proposed by Stütz and Uhl [53]. Their proposed scheme optimizes the quality of the publicly available low quality version by employing JPEG2000 error concealment strategies and encrypts only a small fraction of the JPEG2000 codestream, namely 1–5%. As a consequence the gap in image quality between the publicly available low quality version and a possible attack is reduced.

6 Compression-integrated techniques

Numerous and diverse compression-integrated techniques have been proposed. Encryption in the compression pipeline can be viewed as a compression option (which is kept secret). All considered compression options are not covered in JPEG2000 Part 1, thus their application leads to an encrypted stream not format-compliant with respect to JPEG2000 Part 1.

A major difference among compression-integrated approaches is whether they can be implemented with compliant encoders and decoders. The application of compliant compression software/hardware is an advantage for the practical application of a compression-integrated encryption scheme. Thus the discussion on compression-integrated techniques is divided into two sections, the first discussing techniques that can be implemented with standard compression options, while the second presents various approaches that can only be implemented with non-standard options and with non-standard compression tools.

6.1 Secret standard compression options

The following two approaches aim at using the degrees of freedom in the wavelet transform to construct a unique wavelet domain for the transformation step. By keeping the wavelet domain secret, these approaches provide lightweight security. This procedure can be seen as a form of header encryption, as only the information pertaining to the wavelet domain needs to be encrypted, the rest of the data remains in plaintext. In order to use secret transform domains, Part 2 of the JPEG2000 standard has to be employed. Therefore, a codec that is compliant to JPEG2000 Part 2, is required for encoding and also for decoding of the image in full quality. However, for transparent encryption, a codec compliant to JPEG2000 Part 1, is sufficient to decode the preview image.

6.1.1 Key-dependent wavelet packet subband structures

The wavelet packet decomposition [64] is a generalization of the pyramidal wavelet decomposition, where recursive decomposition may be applied to any subband and is not restricted to the approximation subband. This results in a large space of possible decomposition structures.

Isotropic Wavelet Packets (IWP) Pommer and Uhl [47, 48] propose the use of wavelet packets for providing confidentiality in a zerotree-based wavelet framework. Wavelet packet decompositions are created randomly and kept secret. Engel and Uhl [20] transfer the idea and the central algorithm to JPEG2000 and adapt it to support transparent encryption. The aim for a lightweight encryption scheme with wavelet packets is the definition of a large set of possible bases that perform reasonably well at compression. The process that randomly selects one of the bases from this set should operate in a way that does not give a potential attacker any advantage in an attack. To provide these properties, the construction process is controlled by several parameters, e.g., maximal decomposition depth of certain subbands.

To provide transparent encryption, an additional parameter p is introduced that can be used to optionally specify the number of higher pyramidal resolution levels. If p is set to a value greater than zero, the pyramidal wavelet decomposition is used for resolution levels R_0 through R_p. Non-pyramidal wavelet packets are used for the higher resolution levels, starting from R_{p+1}. With resolution-layer progressions in the final codestream, standard JPEG2000 Part 1 codecs can be used to obtain resolutions R_0 to R_p.

Anisotropic Wavelet Packets (AWP) For the isotropic wavelet packet transform horizontal and vertical decomposition can only occur in pairs. In the anisotropic case this restriction is lifted. The main motivation to introduce anisotropic wavelet packets for lightweight encryption is a substantial increase in keyspace size: the space of possible bases is not only spanned by the decision of decomposing or not, but also by the direction of each decomposition. The amount of data that needs to be encrypted remains extremely small. The complexity of the anisotropic wavelet packet transform is the same as the complexity of the isotropic wavelet packet trans-

form. Like in the isotropic case, compression performance and keyspace size need to be evaluated.

The method for generating randomized wavelet packets has been extended for the anisotropic case by Engel and Uhl [19]. The parameters used to control the generation differ from the isotropic case to reflect the properties of the anisotropic wavelet packet transform. Most notably, the maximum degree of anisotropy is restricted to prevent excessive decomposition into a single direction, as, especially in the case of the approximation subband, this would lead to inferior energy compaction in the wavelet domain for the other direction.

Compression For suitable parameter settings (which facilitate energy compaction, see [19–21]), the average compression performance of the wavelet packet transform is comparable to the performance of the pyramidal wavelet transform.

Security There are two groups of attacks to consider: attacks that try to determine the wavelet packet structure used for encoding, and attacks that try to (partially) reconstruct the transformed image data without knowing the wavelet packet structure.

Reconstruction of Decomposition Structure: Possible attacks that try to determine the wavelet packet structure used for encoding are (a) breaking the cipher with which the decomposition structure was encrypted, (b) inferring the wavelet packet structure from statistical properties of the wavelet coefficients, (c) inferring the wavelet packet structure from the codestream, or (d) performing a full search.

The feasibility of attack (a) is equivalent to the feasibility of breaking the used cipher. Attack (b), inferring the decomposition structure from the codestream tries to use the inclusion metadata in the JPEG2000 codestream. JPEG2000 employs so-called tag trees [55] to signal inclusion information: In a highly contextualized coding scheme, the contributions of each codeblock contained in a packet are linked to the subband structure. Thereby the subband structure is used as context to interpret the output of the tag trees. In order to gather information on either subband structure or coefficients an attacker would have to make a large number of assumptions. However, there are cases (e.g., few quality layers combined with use of markers for packet boundaries) for which fewer possibilities exist and an attacker will have a higher chance of deciphering (some of) the headers. To prevent information leakage, the headers can be encrypted (at the cost of additional computational complexity).

The feasibility of attack (c) is linked to attack (b). If the subband decomposition structure is unknown, the attacker has no way of correctly associating the contributions of a codeblock to the correct coefficients. The attacker therefore lacks full access to the coefficient data (partial access is possible though, see below).

The feasibility of attack (d) depends on the size of the keyspace, which is the number of wavelet packet bases for the used parameters. The number of isotropic wavelet packet bases up to a certain decomposition depth j can be determined recursively, as shown by Xu and Do [69]. Based on this formula, Engel and Uhl [19,21] determine the number of isotropic and anisotropic bases of decomposition level up to j, recursively.

For both, isotropic and anisotropic wavelet packet decompositions, the number of bases obtained with practical parameter settings (i.e., already considering restriction imposed by compression quality requirements) lies above the complexity of a brute-force attack against a 256-bit-key AES cipher.

Partial Reconstruction: Rather than trying to find the used wavelet packet decomposition structure, an attacker can try to partially decode the available data.

For the lower resolutions this approach is successful, prohibiting the use of secret wavelet packet decompositions for full confidentiality. This is due to the fact that the packets of the lowest resolution of any (isotropic) wavelet packet decomposition are the same as the packets produced by a pyramidal decomposition of the same image.

In contrast to encryption for full confidentiality, in a transparent encryption scheme the accessibility of the lower resolutions R_0 (or up to R_p) is desired. Security is only required for the full quality version.

In order to obtain an image of higher quality than R_p, an attacker could try to read a fraction of the coefficient data of R_{p+1} into the pyramidal structure and then attempt a full resolution reconstruction. However, typically the intersection of the randomly generated decomposition structures and the pyramidal structure is far too small to obtain data that allows reconstruction at a substantial quality gain (compared to R_p).

When trying to reconstruct the full quality image, the attacker's problem is how to associate packet data with codeblocks, i.e., spatial location. Again it is the highly contextual coding of JPEG2000 that makes it computationally infeasible for the attacker to correctly perform this association. Engel et al. [16] discuss this issue in more detail.

Performance Wavelet packets bring an increase in complexity as compared to the pyramidal wavelet decomposition: The order of complexity for a level l full wavelet packet decomposition of an image of size N^2 is $\emptyset\left(\sum_{i=1}^{l} 2^{2(i-1)} \frac{N^2}{2^{2(i-1)}}\right)$ compared to $\emptyset\left(\sum_{i=1}^{l} \frac{N^2}{2^{2(i-1)}}\right)$ for the pyramidal decomposition, with the randomized wavelet packet decompositions ranging in-between. With the parameters used in our empirical tests the average time needed for the transform stage increased by 45% as compared to the pyramidal

transform. The average time taken for the whole compression pipeline increased by 25%.

The anisotropic wavelet packet transform does not increase complexity compared to the isotropic case. As more bases can be constructed with lower decomposition depths, the use of the anisotropic wavelet packet transform lowers the computational demands of the scheme.

In general, wavelet packets dramatically reduce the effort for encryption compared to full encryption and other partial or selective encryption schemes. This circumstance makes encryption with a public key scheme feasible, which reduces the effort for key management considerably.

However, the considerable computational complexity that is introduced for the transform step needs to be taken into account for potential application scenarios. For some application scenarios the decrease of complexity in the encryption stage might not suffice to justify the increase of complexity in the compression stage.

6.1.2 Parameterized lifting schemes

Three wavelet parameterization schemes have been investigated in the context of lightweight encryption: the parameterization for a family of orthogonal wavelets proposed by Schneid and Pittner [50], the parameterization for even and odd length biorthogonal filters proposed by Hartenstein et al. [26], and the lifting parameterization of the CDF 9/7 wavelet proposed by Zhong and Jiao [71]. Köckerbauer and Uhl [36] report that in the context of JPEG2000 the first parameterization produces unreliable compression results.

Engel and Uhl [17] use the biorthogonal lifting parameterization presented by Zhong and Jiao [71] with JPEG2000 and report compression performance that is superior to the other parameterization schemes. The used parameterization constructs derivations of the original CDF 9/7 wavelet based on a single parameter α.

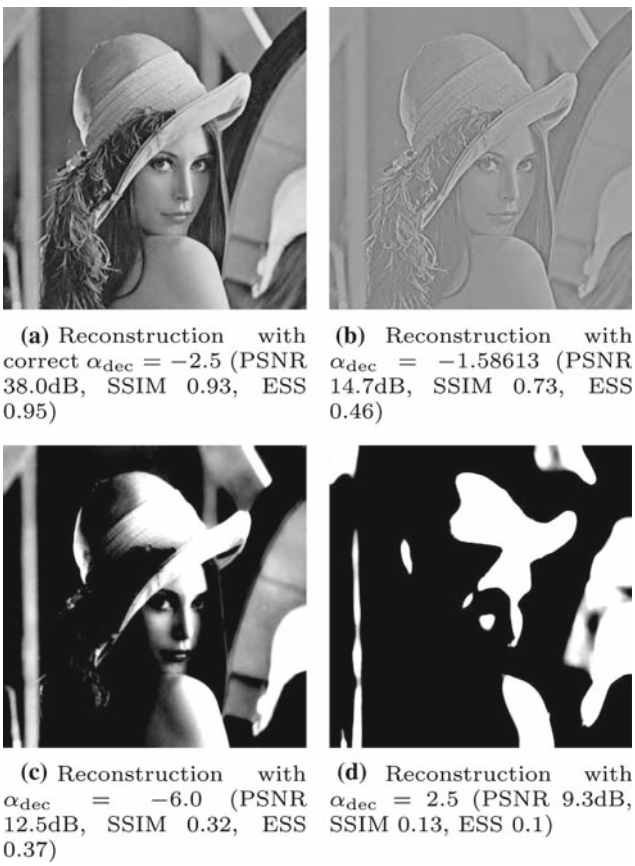
Compression Compression performance of the produced filters and their utility for JPEG2000 lightweight encryption are investigated in [17]. The tests show that the range in which the parameterized filters achieve good compression results on the one hand and exhibit sufficient variation to withstand a brute-force attack is rather limited. In [18], Engel and Uhl argue that, because filters vary much more for lower absolute values of α, discretization bins should not be uniform. In order to enlarge the keyspace, they further propose to use different parameters for the horizontal and the vertical wavelet decomposition on different decomposition levels. These techniques have been called "non-stationary" (varying on each decomposition level) and "inhomogeneous" (varying in vertical and horizontal orientation) in the context of adaptive compression [58]. Neither of these methods results in a significant deterioration of compression performance.

The distortion introduced by this scheme is a loss of luminance information rather than a loss of structural information. Figure 10 shows some examples of reconstructed version of the Lena image encrypted with parameter $\alpha = 2.5$: (a) shows the image reconstructed with the correct parameter, (b), (c) and (d) show the image reconstructed with incorrect parameters.

Security *Brute-force:* It is reported in [18] that for keys of small individual values for all parameters in each direction and each level a brute-force search for the full-quality version remains unsuccessful. However, keys exist where each parameter is of higher absolute value for which the brute-force attack comes close to the full quality version.

Symbolic Attack: A principal attack on parameterized lifting schemes is presented by Engel et al. in [14]. It is based on the symbolic computation of the inverse wavelet transform.

An attacker, who does not know the parameter values for the parameterized transform, can build a symbolic expression for each pixel value in the reconstructed image containing the necessary operations for the inverse transformation.



(a) Reconstruction with correct $\alpha_{\text{dec}} = -2.5$ (PSNR 38.0dB, SSIM 0.93, ESS 0.95)

(b) Reconstruction with $\alpha_{\text{dec}} = -1.58613$ (PSNR 14.7dB, SSIM 0.73, ESS 0.46)

(c) Reconstruction with $\alpha_{\text{dec}} = -6.0$ (PSNR 12.5dB, SSIM 0.32, ESS 0.37)

(d) Reconstruction with $\alpha_{\text{dec}} = 2.5$ (PSNR 9.3dB, SSIM 0.13, ESS 0.1)

Fig. 10 Parameterized wavelet filters: reconstructed images and quality measure results for the Lena image ($\alpha_{\text{enc}} = -2.5$), rate 1 bpp

The resulting term will depend on the transform coefficients, which are known to the attacker. The only unknowns are formed by the parameters of the transform. By performing a full symbolic inverse wavelet transformation, the attacker can construct a complete symbolic description of the operations necessary to reconstruct the plaintext image.

A ciphertext only attack in this context remains largely unsuccessful. This is due to the lack of a reliable non-reference image quality metric.

Known-plaintext attacks are much more successful. If the full plaintext is known, then the symbolic representation can be used to determine the used parameters. This also works if more parameters are used (as in the case of inhomogeneous and non-stationary variation).

Also if only partial plaintext information is available, the symbolic representation yields successful attacks.

Engel et al. [14] discuss two possible scenarios in this context: For the pixel samples attack, the attacker is assumed to have obtained individual pixel samples from the reconstructed image; for the average luminance value attack only the average luminance value from a preview image is required. For both cases, the attacker can obtain a more or less accurate solution for the used wavelet parameters.

Inhomogeneous and non-stationary variation as well as higher-dimensional parameterizations of the wavelet transform increase the number of parameters and therefore make the attack more difficult. However, on a principal note, these symbolic attacks show a general problem of lightweight encryption schemes that rely on linear transforms for providing security. Such attacks severely compromise the security of encryption schemes that use a parameterized wavelet transform, even if their claim is to provide only lightweight encryption.

Performance The filter parameterization comes at virtually no cost: Apart from five values in the lifting scheme that have to be computed for each used parameter value, no additional complexity is introduced.

6.2 Secret non-standard compression options

Many proposals for compression-integrated encryption modify parts of the compression pipeline in a non-standardized fashion.

6.2.1 Wavelet coefficient sign encryption

The signs of the wavelet coefficients are scrambled in a number of contributions [9–13], mainly with the goal to preserve privacy. In [11], as well as in [10], flipping the signs of selected coefficients is proposed for "privacy enabling technology for video surveillance". This scheme may also be applied selectively in the transform domain, scrambling only parts of the image.

Compression There is only a small negative influence on compression performance. According to [10] the bitrate is increased by less than 10% bitrate (i.e., less than 1dB for a wide range of compression ratios).

Security The pseudo-random flipping of wavelet coefficient signs may be subject to specific cryptoanalysis. In [49] Said shows the insecurity of DCT sign encryption; however, he uses strong assumptions for his cryptoanalytic framework as well as for his attack.

Performance The introduced overhead is negligible [10].

6.2.2 Random permutations

Norcen and Uhl [43,44] have investigated the usage of random permutations applied to wavelet coefficients within the JPEG2000 coding pipeline. Both confidential and transparent encryption can be implemented by applying permutations to the appropriate subbands, however, one has to keep the inherent security concerns regarding permutations in mind (e.g., vulnerability against known plaintext attacks).

Norcen and Uhl [43] have investigated the permutation of single coefficients within wavelet subbands. In this approach the compression performance is degraded significantly, because the intra-subband dependencies of the coefficients are destroyed. They show that a key generation algorithm has to be employed, since the direct embedding of the permutation key is not feasible from a compression point of view.

In later work [44] aim at improving the rate-distortion performance of permutation based schemes by permuting and rotating differently sized blocks of coefficients (instead of single coefficients) within wavelet subbands. The best compromise with respect to the tradeoff between compression performance and security turns out to be the *blockwise-fully-adaptive* scheme where each subband is divided into the same number of blocks (e.g., 64) which are then permuted. Additionally to the permutation on a block basis, the blocks can be rotated, which increases the keyspace but does not influence compression quality.

Compression Compression performance may suffer from the destruction of coefficient statistics and cross correlations through permutation and rotation. In [43] it has been shown that permutation applied to single coefficients severely reduces the compression performance (up to 35%). Schemes applying permutations to blocks of coefficients have been found to be more suited with respect to compression quality [44]—the image quality is augmented with increasing blocksize, however, security is decreased with increasing blocksize

(see below). In the blockwise-fully-adaptive scheme compression performance loss can be kept below 10%.

Security The security of the presented permutation schemes strongly relies on the blocksize used. Basically there are $n!$ permutations of n elements, in this case, coefficients or blocks of coefficients. The keyspace of a specific subband thus is $n!$ where n is the number of its blocks (or single coefficients). The whole keyspace is the product of the keyspaces of all subbands. For the block-based permutation the keyspace for a certain subband is $(width \times height/blocksize^2)!$. If additionally a random rotation is applied, then the keyspace of a certain subband is $4^b b!$, where b is the number of blocks. For the blockwise-full-adaptive case each subband has 64! different permutations. If random rotation is applied, this number is increased to $4^{64}64!$ (except if the remaining block consists only of one coefficient).

In fact not all of the blocks are different (in the high frequency subbands zero coefficients are very likely). If k blocks are similar, then the number of permutations is decreased by $k!$ For the blockwise-full-adaptive scheme for 15 similar blocks there are still $64!/15! = 2 \times 10^{26}$ possible permutations.

However, the security of a system is not entirely determined by its keyspace. Keeping in mind that the lower subbands contain the visually most important information, it has to be pointed out that those are naturally secured by a smaller keyspace.

Moreover, the actual strength of the permutation approach is reduced since correlations among neighboring block borders can be exploited. Hence the bigger the blocks, the less secure the scheme.

An image with permuted 16×16 blocks reveals a considerable amount of image information, mostly due to the fact that the lowest resolution subband is not modified at all (it contains exactly one 16×16 block). In general the problem with fixed size permutations is that the visually more important subbands are not better secured, hence the full keyspace is a wrong assumption, because an attacker might be able to deduce information from the lower subbands without even considering the higher frequency parts. This is especially true if the blocksizes are in excess of 16×16, which mostly leads to unencrypted low frequency subbands.

In the blockwise-fully-adaptive scheme, the number of blocks can be adjusted to a certain security level and the problems with fixed sized blocks are resolved.

A permutation per coefficient destroys all block correlations and can therefore be considered the most secure type of permutation. As a consequence there is a trade-off between security and compression performance.

Another important aspect is information leakage. Since the wavelet coefficients are not changed with this encryption scheme, a simple comparison between the coefficients of an assumed plaintext image and that of the encrypted image will reveal its identity.

Performance The entire compression pipeline has to be run through, but the additional effort is negligibly small (according to our experimental tests).

6.2.3 Mixed perturbations

Lian et al. [37,38] propose the combination of several compression-integrated encryption schemes, such as sign encryption of the wavelet coefficients, inter block permutation and bitplane permutation. Additionally they introduce a parameter q, the quality factor ranging from 0 to 100, to adjust the encryption strength and the actual image quality of a reconstruction. Hence their scheme may be employed to implement transparent encryption among other application scenarios. In more detail, the quality factor determines the percentage of coefficients for which sign encryption is conducted (for a quality factor of 0 the signs of all coefficients are encrypted, while for a quality factor of 100 no sign is encrypted), the number of intra-permuted codeblocks (bitplane permutation) and a boolean decision whether inter block permutation is employed (which is conducted on a codeblock basis). The order in which both codeblocks and coefficients are treated is from high frequency to low frequency and thus the quality decreases rather smoothly with a decrease of the quality factor.

Compression The compression ratio is reduced. An example is given where the degradation is less than 1.5dB for all bitrates in [37]. To put the loss of compression performance into context, JPEG2000 outperforms JPEG by about 2.5dB (PSNR) for a wide range of compression ratios (for the well-known Lena image of size 512×512 pixels).

Security There are no known attacks against this scheme; however, every single perturbation may be subject to specific cryptoanalysis.

Performance For a quality factor of 0 (lowest quality) the encryption process takes 7.5–13.2% (as reported in [37]) of the compression (details about the applied software and parameters are not known).

6.2.4 Randomized arithmetic coding (RAC)

Grangetto et al. [24] propose JPEG2000 encryption by randomized arithmetic coding. Although the arithmetic coder of the JPEG2000 pipeline is altered, their approach has no influence on the compression performance. The basic idea of their approach is to change the order of the probability intervals in the arithmetic coding process. For the partitioning of

the probability interval, it is a convention (agreed upon by both the encoder and the decoder) which interval (either that of the most probable or that of the least probable symbol) is the preceding one. In [24], for every encoded decision bit the ordering of the intervals is chosen securely randomly (by using a random bit from the PRNG).

Selective/partial application of this encryption approach is possible.

Compression There is no influence on compression performance.

Security Packet header information is left unencrypted and thus the same considerations as for packet body based format-compliant encryption schemes apply (cf. Sects. 5.3.8). An entire section in [24] is dedicated to the cryptoanalysis of their method. It is noted that their method might be susceptible to known-plaintext attacks, but it is argued that these kinds of attacks are not relevant for the proposed encryption systems. A possible counter-argument to this assumption is that, as codeblocks in higher frequency subbands tend to be quantized to zero, it is likely that compressed codeblock contributions of higher frequencies represent bitplanes with a vast majority of zeros.

Due to performance issues, Grangetto et al. propose the usage of a weaker PRNG (with a 32 bit key) based on the standard rand function of the Linux C library. The analysis of the security of this PRNG is out of the scope of this paper, but it can be considered a possible vulnerability. A key size of 32 bit is too short for serious security anyway. Alternatively, the secure random number generator proposed in [3] is employed. However, more secure and efficient PRNG can be considered, e.g., AES in OFB mode.

Performance The entire compression pipeline has to be run through, the additional effort arises from the intensive usage of the PRNG. For every decision bit (one per coefficient and per bitplane) coded in the arithmetic coder, a random bit is required. This amount of randomness (basically the same as for raw encryption) induces the authors to employ a faster random number generator (encryption time of 0.33 s for the Lena image with 512×512 pixel). Using a secure random number generator [3], the authors report an encryption time of 370.01 s for the full encryption of the Lena image with 512×512 pixel. However, the usage of such a computationally complex PRNG is not justified and instead AES in OFB mode could be used as PRNG. Employing our implementation of the AES OFB mode, the generation of the pseudo random keystream of the appropriate length for the Lena image with 512×512 pixel only takes 0.045 s. Thus even for secure settings the increase in complexity is not that exorbitant.

However, the computational complexity of this approach is high.

6.2.5 Secret initial tables in the MQ-coder

This approach has been proposed by Liu [39] and as in the previously discussed approach of RAC (see Sect. 6.2.4), the entropy coding stage is modified. The arithmetic coding engine (the MQ-coder) receives a context label and a decision (MPS, more probable symbol or LPS, less probable symbol). There are 19 context labels in JPEG2000. The estimation of current interval size for a context is conducted via a finite state machine with 47 states. At the start of entropy coding, each context label is assigned an initial state [34, p. 89]. Liu proposes to randomly select these initial states in order to prevent standard JPEG2000 decoders from correctly decoding the data.

Like the RAC approach, this approach is closely related to packet body encryption with bitstream-compliant algorithms. Selective/partial application of this encryption approach is possible as well.

Compression According to [39] the compression overhead is negligible (compressibility equivalent).

Security Packet header information is left unencrypted and thus the same considerations as for packet body based format-compliant encryption schemes apply (cf. Sects. 5.3.8).

According to [39] the approach is computationally secure as there are 47^{19} (approx. $2^{105.5}$) possible initial tables. However, a huge key space may not prevent specifically tailored attacks against this scheme.

Performance The computational complexity remains almost the same as for the standard JPEG2000 Part 1 compression pipeline; the only effort is to build the random initial table.

7 Discussion and overview

In this Section, we will provide a general discussion on which techniques are appropriate for the different application scenarios discussed in the introduction.

The naive encryption technique, i.e., encrypting the entire JPEG2000 codestream with a classical cipher, of course achieves a higher data throughput as compared to all format-compliant bitstream-oriented techniques at the highest level of security. Additionally, information leakage occurs neither for header information nor for packet data. Therefore, if format compliance and all associated functionalities that rely on the JPEG2000 codestream structure are not an issue in the target application, naive encryption is the method

of choice (e.g., as in the DCI security scheme discussed in Sect. 3.6.1).

The discussed format-compliant bitstream-oriented techniques can meet the demands of both on-line and off-line scenarios. Furthermore, an almost arbitrary range of confidentiality levels may be supported by employing partial/selective encryption, ranging from transparent/perceptual encryption where even a certain quality of the visual data in encrypted form has to be guaranteed, to sufficient encryption where strict security is not the major goal but only pleasant viewing has to be impossible. Of course, also high security scenarios may be supported by simply encrypting all the packet data and/or even packet header data. These facts taken together make format-compliant bitstream-oriented encryption techniques the most flexible and most generally applicable schemes discussed.

Considering all approaches including segment based encryption, the KLV approach of the DCI standard and of course all format-compliant encryption schemes one has to mention that a small fingerprint of the image (the compressed size) is preserved by all. For a single image this information can be regarded as insignificant, however, a series of these fingerprints, e.g., obtained from an encrypted movie, identifies the source data in a rather unique way. (Of course the identification only works, if the rate is dynamically adjusted.)

Compression-integrated techniques can only be applied in a sensible way in on-line application scenarios. When compared to bitstream-oriented techniques, the computational demand for encryption is significantly reduced, however, the reduction of complexity for encryption comes at the cost of a (more or less significant) rise in complexity in the compression pipeline. In all schemes considered the impact on compression performance can be kept to a minimum if applied with care.

In a certain sense, the use of key-dependent wavelet transforms in encryption is an extreme case of selective/partial encryption since encryption is limited to the actual subband structure/filter choice. The corresponding amount of data to be encrypted is so small, that this approach can directly employ public key encryption schemes and benefit from their superior key management. Another advantage is that because even though the coefficient data cannot be interpreted without the correct transform at hand, it can be used to perform signal processing in the encrypted domain. For example, this can be useful in a retrieval-based application for creating hashes of encrypted visual data, which facilitates search in the encrypted domain. With respect to the level of confidentiality that can be supported, both wavelet packet and parameterized filters based schemes are found to be restricted to the transparent/perceptual (potentially also to the sufficient) encryption application scenario—real privacy cannot be achieved. Whereas the increase in complexity that is shifted to the compression pipeline can be considered significant for the wavelet packet case, in the case of parameterized filters there is only negligible additional cost. The successful attacks against the approach based on parameterized filters renders this technique almost useless in environments where sincere attacks are to be expected. At most, in settings that require soft encryption, e.g., in the area of mobile multimedia applications, the level of security might suffice and the extremely low computational demands could be an incentive for using parameterized wavelet filters. It has to be pointed out that the low amount of encryption required for transparent encryption in key-dependent transform techniques makes those especially attractive since the classical approach for transparent encryption in bitstream-oriented techniques requires almost the entire codestream to be encrypted (compare Sect. 5.5). However, more recent techniques [53] only require a fraction of the encryption amounts. Engel et al. [16] discuss various application scenarios for transparent encryption where one or the other approach might be of advantage.

Finally, employing permutations within the JPEG2000 pipeline is a classical case of soft encryption. The computational overhead remains negligible, however, permutations are of course vulnerable to known plaintext attacks unless the keys are exchanged frequently. Contrasting to the previous techniques also higher levels of confidentiality may be targeted (this just depends on which subbands are subject to permutations), however, the security flaws present with permutations should be kept in mind.

Information leakage is significantly higher in compression-integrated techniques as compared to bitstream-oriented ones. The entire coefficient information is available in plaintext—due to the missing context this cannot be exploited for reconstructing the original data, but all sorts of statistical analyses may be conducted on these data potentially allowing an attacker to identify an image in encrypted form. Therefore, the security level of for these compression-integrated schemes has to be assessed to be lower as compared to bitstream-oriented ones in general.

A completely different approach is randomized arithmetic coding as proposed by Grangetto et al. It is closely linked to the bitstream-compliant encryption approaches discussed in Sect. 5 as it targets the coefficient data contained in the packet bodies. However, the drawbacks of this solution are the increased complexity compared to bitstream-compliant approaches. If bitstream-compliant approaches are applied on a CCP basis there is no difference in the preserved functionality (given the appropriate key management for both schemes). Thus the randomized arithmetic coding approach has the same functionality as the bitstream-compliant approaches, but obvious disadvantages. Secret initial tables may be an interesting option; however, the security of this approach against specifically tailored attacks remains to be proven. The selective/partial application cannot gain substantial performance gains, as the encryption only takes a

Table 1 An overview of JPEG2000 encryption approaches

Approach	Naive	Segment-based	Bitstream-compliant	Wavelet packets	Filters	Permutations	RAC
Compression decrease	None	Slightly	None	Moderately	Slightly	Moderately	None
Confidentiality	Privacy	All levels (ZOI)	All levels (ZOI)	Transparent	Transparent	All levels (ZOI)	All levels (ZOI)
Security	Very high	High–very high	High	Medium–high	Low	Medium	Medium–high
Transcodability	None	Partial, segment based	On packet basis	JPEG2000	JPEG2000	JPEG2000	JPEG2000
Transcode complexity	Very high	Very low	Very low with markers	JPEG2000	JPEG2000	JPEG2000	JPEG2000
Create complexity	Low	Low–medium	Low	High*	Very low–low*	Very low*	High*
Compression pipeline	Full	Tier2	No	Full	Full	Full	Full
Consume complexity	Low	Low–medium	Low	High	Very low–low	Very low	Medium
Uninformed consume	Impossible	Impossible	Possible	Possible	Possible	Possible	Possible
Error propagation	Avalanche eff.	Within segment	Mostly JPEG2000	JPEG2000	JPEG2000	JPEG2000	JPEG2000

negligible fraction of the entire compression and encryption system (cf. Sect. 5.3.8).

In Table 1, we provide a concise summary of the various aspects discussed in this and the preceding Sections. A "*" indicates that the entire compression pipeline has to be conducted and the given information is with respect to the additional effort within the compression pipeline.

8 Conclusion

In this survey we have discussed and compared various techniques for protecting JPEG2000 codestreams by encryption technology. As to be expected, some techniques turn out to be more beneficial than others and some methods hardly seem to make sense in any application context. In any case, a large variety of approaches exhibiting very different properties can be considered useful and covers almost any thinkable multimedia application scenario. This survey provides a guide to find the proper JPEG2000 encryption scheme for a target application.

References

1. Apostolopoulos, J., Wee, S., Dufaux, F., Ebrahimi, T., Sun, Q., Zhang, Z.: The emerging JPEG2000 security (JPSEC) standard. In: Proceedings of International Symposium on Circuits and Systems, ISCAS'06. IEEE, May 2006
2. Apostolopoulos, J.G., Wee S.J: Supporting secure transcoding in JPSEC. In: Tescher, A.G. (ed.) Applications of Digital Image Processing XXVIII, vol. 5909, p. 59090J. SPIE (2005)
3. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo-random number generator. SIAM J. Comput. **15**(2), 364–383 (1986)
4. Bradley, J.N., Brislawn, C.M., Hopper, T.: The FBI wavelet/scalar quantization standard for gray-scale fingerprint image compression. In: SPIE Proceedings, Visual Information Processing II, vol. 1961, pp. 293–304, Orlando, FL, USA, April (1993)
5. Conan, V., Sadourny, Y., Jean-Marie, K., Chan, C., Wee, S., Apostolopoulos, J.: Study and validation of tools interoperability in JPSEC. In: Tescher, A.G. (ed.) Applications of Digital Image Processing XXVIII, vol. 5909, p. 59090H. SPIE (2005)
6. Conan, V., Sadourny, Y., Thomann, S.: Symmetric block cipher based protection: Contribution to JPSEC. ISO/IEC JTC 1/SC 29/WG 1 N 2771 (2003)
7. Daemen, J., Rijmen, V.: The Design of Rijndael: AES—The Advanced Encryption Standard. Springer, Berlin (2002)
8. Digital Cinema Initiatives, LLC (DCI). Digital cinema system specification v1.2. online presentation, March (2008)
9. Dufaux, F., Ebrahimi, T.: Region-based transform-domain video scrambling. In: Proceedings of Visual Communications and Image Processing, VCIP'06. SPIE (2006)
10. Dufaux, F., Ebrahimi, T.: Scrambling for video surveillance with privacy. In Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop, CVPRW '06. IEEE (2006)
11. Dufaux, F., Ouaret, M., Abdeljaoued, Y., Navarro, A., Vergnenegre, F., Ebrahimi, T.: Privacy enabling technology for video surveillance. In: Proceedings of SPIE, Mobile Multimedia/Image Processing for Military and Security Applications, vol. 6250. SPIE (2006)
12. Dufaux, F., Wee, S., Apostolopoulos J., Ebrahimi T.: JPSEC for secure imaging in JPEG2000. In: Tescher, A.G. (ed.) Applications of Digital Image Processing XXVII, vol. 5558, pp. 319–330. SPIE (2004)
13. Dufaux, F., Ebrahimi, T.: Securing JPEG2000 compressed images. In: Tescher, A.G. (ed.) Applications of Digital Image Processing XXVI, vol. 5203, pp. 397–406. SPIE (2003)
14. Engel, D., Kutil, R., Uhl, A.: A symbolic transform attack on lightweight encryption based on wavelet filter parameterization. In: Proceedings of ACM Multimedia and Security Workshop, MM-SEC '06, pp. 202–207, Geneva, Switzerland, September (2006)
15. Engel, D., Stütz, T., Uhl, A.: Format-compliant JPEG2000 encryption in JPSEC: Security, applicability and the impact of compression parameters. EURASIP J. Inform. Secur. (Article ID 94565), 20 (2007). doi:10.1155/2007/94565
16. Engel, D., Stütz, T., Uhl, A.: Efficient transparent JPEG2000 encryption. In: Li, C.-T. (ed.) Multimedia Forensics and Security, pp. 336–359. IGI Global, Hershey, PA, USA (2008)

17. Engel, D., Uhl, A.: Parameterized biorthogonal wavelet lifting for lightweight JPEG2000 transparent encryption. In: Proceedings of ACM Multimedia and Security Workshop, MM-SEC '05, pp. 63–70, New York, NY, USA, August (2005)
18. Engel, D., Uhl, A.: Security enhancement for lightweight JPEG2000 transparent encryption. In: Proceedings of Fifth International Conference on Information, Communication and Signal Processing, ICICS '05, pp. 1102–1106, Bangkok, Thailand, December (2005)
19. Engel, D., Uhl, A.: Lightweight JPEG2000 encryption with anisotropic wavelet packets. In: Proceedings of International Conference on Multimedia and Expo, ICME '06, pp. 2177–2180, Toronto, Canada, July 2006. IEEE (2006)
20. Engel, D., Uhl, A.: Secret wavelet packet decompositions for JPEG2000 lightweight encryption. In: Proceedings of 31st International Conference on Acoustics, Speech, and Signal Processing, ICASSP '06, vol. V, pp. 465–468, Toulouse, France, May 2006. IEEE (2006)
21. Engel, D., Uhl, A.: An evaluation of lightweight JPEG2000 encryption with anisotropic wavelet packets. In: Delp, E.J., Wong, P.W. (eds.) Security, Steganography, and Watermarking of Multimedia Contents IX. Proceedings of SPIE, pp. 65051S1–65051S10, San Jose, CA, USA, January 2007. SPIE (2007)
22. Fang, J., Sun, J.: Compliant encryption scheme for JPEG2000 image code streams. J. Electron. Imaging **15**(4) (2006)
23. Furht, B., Kirovski, D. (eds.): Multimedia Security Handbook. CRC Press, Boca Raton, (2005)
24. Grangetto, M., Magli, E., Olmo, G.: Multimedia selective encryption by means of randomized arithmetic coding. IEEE Trans. Multimed. **8**(5), 905–917 (2006)
25. Grosbois, R., Gerbelot, P., Ebrahimi, T.: Authentication and access control in the JPEG2000 compressed domain. In: Tescher, A.G. (ed.) Applications of Digital Image Processing XXIV. Proceedings of SPIE, vol. 4472, pp. 95–104, San Diego, CA, USA, July (2001)
26. Hartenstein, F.: Parametrization of discrete finite biorthogonal wavelets with linear phase. In: Proceedings of the 1997 International Conference on Acoustics, Speech and Signal Processing (ICASSP'97), April (1997)
27. Imaizumi, S., Watanabe, O., Fujiyoshi, M., Kiya, H.: Generalized hierarchical encryption of JPEG2000 codestreams for access control. In: Proceedings of the IEEE International Conference on Image Processing (ICIP'05), vol. 2. IEEE, September (2005)
28. Imaizumi, S., Fujiyoshi, M., Abe, Y., Kiya, H.: Collusion attack-resilient hierarchical encryption of JPEG 2000 codestreams with scalable access control. In: Image Processing, 2007. ICIP 2007. IEEE International Conference on, vol. 2, pp. 137–140, September (2007)
29. ISO/IEC 15444-8, Final Committee Draft. Information technology—JPEG2000 image coding system, Part 8: Secure JPEG2000. Technical report, ISO, November (2004)
30. ISO/IEC 15444-1. Information technology—JPEG2000 image coding system, Part 1: Core coding system, December (2000)
31. ISO/IEC 15444-4. Information technology—JPEG2000 image coding system, Part 4: Conformance testing, December (2004)
32. ISO/IEC 15444-8. Information technology—JPEG2000 image coding system, Part 8: Secure JPEG2000, April (2007)
33. ITU-T H.264. Advanced video coding for generic audivisual services, November (2007)
34. ITU-T T.800. Information technology—JPEG2000 image coding system, Part 1: Core coding system, August (2002)
35. Kiya, H., Imaizumi, D., Watanabe O.: Partial-scrambling of image encoded using JPEG2000 without generating marker codes. In: Proceedings of the IEEE International Conference on Image Processing (ICIP'03), vol. III, pp. 205–208, Barcelona, Spain, September (2003)
36. Köckerbauer, T., Kumar, M., Uhl, A.: Lightweight JPEG2000 confidentiality for mobile environments. In: Proceedings of the IEEE International Conference on Multimedia and Expo, ICME '04, Taipei, Taiwan, June (2004)
37. Lian, S., Sun, J., Wang, Z.: Perceptual cryptography on JPEG2000 compressed images or videos. In: 4th International Conference on Computer and Information Technology, Wuhan, China, September 2004. IEEE (2004)
38. Lian, S., Sun, J., Zhang, D., Wang, Z.: A selective image encryption scheme based on JPEG2000 codec. In: Nakamura, Y., Aizawa, K., Satoh, S. (eds.) Proceedings of the 5th Pacific Rim Conference on Multimedia. Lecture Notes in Computer Science, vol. 3332, pp. 65–72. Springer, Berlin (2004)
39. Liu, J.-L.: Efficient selective encryption for jpeg 2000 images using private initial table. Pattern Recognit. **39**(8), 1509–1517 (2006)
40. Lo, S.-C.B., Li, H., Freedman, M.T.: Optimization of wavelet decomposition for image compression and feature preservation. IEEE Trans. Med. Imaging **22**(9), 1141–1151 (2003)
41. Macq, B.M., Quisquater, J.-J.: Cryptology for digital TV broadcasting. Proc. IEEE **83**(6), 944–957 (1995)
42. Mao, Y., Wu, M.: Security evaluation for communication-friendly encryption of multimedia. In: Proceedings of the IEEE International Conference on Image Processing (ICIP'04), Singapore, October 2004. IEEE Signal Processing Society (2004)
43. Norcen, R., Uhl, A.: Encryption of wavelet-coded imagery using random permutations. In: Proceedings of the IEEE International Conference on Image Processing (ICIP'04), Singapore, October 2004. IEEE Signal Processing Society (2004)
44. Norcen, R., Uhl, A.: Performance analysis of block-based permutations in securing JPEG2000 and SPIHT compression. In: Li, S., Pereira, F., Shum ,H.-Y., Tescher, A.G. (eds.) Visual Communications and Image Processing 2005 (VCIP'05). SPIE Proceedings, vol. 5960, pp. 944–952, Beijing, China, July 2005. SPIE (2005)
45. Norcen, R., Uhl, A.: Selective encryption of the JPEG2000 bitstream. In: Lioy, A., Mazzocchi, D. (eds.) Communications and Multimedia Security. Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security, CMS '03. Lecture Notes on Computer Science, vol. 2828, pp. 194–204, Turin, Italy, October 2003. Springer, Berlin (2003)
46. Pommer, A., Uhl, A.: Application scenarios for selective encryption of visual data. In: Dittmann, J., Fridrich, J., Wohlmacher, P. (eds.) Multimedia and Security Workshop, ACM Multimedia, pp. 71–74, Juan-les-Pins, France, December (2002)
47. Pommer, A., Uhl, A.: Selective encryption of wavelet packet subband structures for secure transmission of visual data. In: Dittmann, J., Fridrich, J., Wohlmacher, P. (eds) Multimedia and Security Workshop, ACM Multimedia. pp. 67–70, Juan-les-Pins, France, December (2002)
48. Pommer, A., Uhl, A.: Selective encryption of wavelet-packet encoded image data—efficiency and security. ACM Multimed. Syst. (Special Issue Multimed. Secur.) **9**(3), 279–287 (2003)
49. Said, A.: Measuring the strength of partial encryption schemes. In: Proceedings of the IEEE International Conference on Image Processing (ICIP'05), vol. 2, September (2005)
50. Schneid, J., Pittner, S.: On the parametrization of the coefficients of dilation equations for compactly supported wavelets. Computing **51**, 165–173 (1993)
51. Schneier, B.: Applied cryptography: protocols, algorithms and source code in C, 2nd edn. Wiley, New York (1996)
52. Stütz, T., Uhl, A.: On format-compliant iterative encryption of JPEG2000. In: Proceedings of the Eighth IEEE International Symposium on Multimedia (ISM'06), pp. 985–990, San Diego, CA, USA, December 2006. IEEE Computer Society (2006)
53. Stütz, T., Uhl, A.: On efficient transparent JPEG2000 encryption. In Proceedings of ACM Multimedia and Security Workshop,

MM-SEC '07, pp. 97–108, New York, NY, USA, September 2007. ACM Press
54. Stütz, T., Uhl, A.: On JPEG2000 error concealment attacks. In: Proceedings of the 3rd Pacific-Rim Symposium on Image and Video Technology, PSIVT '09, Lecture Notes in Computer Science, Tokyo, Japan, January 2009. Springer, Berlin (2009, to appear)
55. Taubman, D.: High performance scalable image compression with EBCOT. IEEE Trans. Image Process. **9**(7), 1158–1170 (2000)
56. Taubman, D., Marcellin, M.W.: JPEG2000—Image Compression Fundamentals, Standards and Practice. Kluwer, Dordrecht (2002)
57. Tolba, A.S.: Wavelet packet compression of medical images. Digit. Signal Process. **12**(4), 441–470 (2002)
58. Uhl, A.: Image compression using non-stationary and inhomogeneous multiresolution analyses. Image Vis. Comput. **14**(5), 365–371 (1996)
59. Uhl, A., Obermair, Ch.: Transparent encryption of JPEG2000 bitstreams. In: Podhradsky, P. et al. (eds.) Proceedings EC-SIP-M 2005 (5th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services), pp 322–327, Smolenice, Slovak Republic (2005)
60. Uhl, A., Pommer, A.: Image and Video Encryption. From Digital Rights Management to Secured Personal Communication. Advances in Information Security, vol. 15. Springer, Berlin (2005)
61. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4) (2004)
62. Wee, S., Apostolopoulos, J.: Secure transcoding with JPSEC confidentiality and authentication. In: Proceedings of the IEEE International Conference on Image Processing (ICIP'04), Singapore, Singapore, October (2004)
63. Wee, S.J., Apostolopoulos, J.G.: Secure scalable streaming and secure transcoding with JPEG2000. In: Proceedings of the IEEE International Conference on Image Processing (ICIP'03), vol. I, pp. 547–551, Barcelona, Spain, September (2003)
64. Wickerhauser, M.V.: Adapted Wavelet Analysis from Theory to Software. A.K. Peters, Wellesley (1994)
65. Wu, H., Ma, D.: Efficient and secure encryption schemes for JPEG2000. In: Proceedings of the 2004 International Conference on Acoustics, Speech and Signal Processing (ICASSP 2004), pp. 869–872, May (2004)
66. Wu, M., Mao, V.: Communication-friendly encryption of multimedia. In: Proceedings of the IEEE Multimedia Signal Processing Workshop, MMSP '02, St. Thomas, Virgin Islands, USA, December (2002)
67. Wu, Y., Deng, R., Ma, D.: Securing JPEG2000 code-streams. In: Lee, D., Shieh, S., Tygar, J., (eds.) Computer Security in the 21st Century, pp. 229–254 (2005)
68. Wu, Y., Deng, R.H.: Compliant encryption of JPEG2000 codestreams. In: Proceedings of the IEEE International Conference on Image Processing (ICIP'04), Singapure, October 2004. IEEE Signal Processing Society (2004)
69. Xu, D., Do, M.N.: Anisotropic 2-D wavelet packets and rectangular tiling: theory and algorithms. In: Unser, M.A., Aldroubi, A., Laine, A.F. (eds.) Proceedings of SPIE Conference on Wavelet Applications in Signal and Image Processing X. SPIE Proceedings, vol. 5207, pp. 619–630, San Diego, CA, USA, August 2003. SPIE (2003)
70. Yang, Y., Zhu, B.B., Yang, Y., Li, S., Yu N.: Efficient and syntax-compliant JPEG2000 encryption preserving original fine granularity of scalability. EURASIP J. Inform. Secur. (2007)
71. Zhong, G., Cheng, L., Chen, H.: A simple 9/7-tap wavelet filter based on lifting scheme. In: Proceedings of the IEEE International Conference on Image Processing (ICIP'01), pp. 249–252, October (2001)
72. Zhu, B., Yang, Y., Li, S.: JPEG2000 syntax-compliant encryption preserving full scalability. In: Proceedings of the IEEE International Conference on Image Processing (ICIP'05), vol. 3, September (2005)