Hindawi Publishing Corporation
EURASIP Journal on Information Security
Volume 2007, Article ID 94565, 20 pages
doi:10.1155/2007/94565

Research Article

Format-Compliant JPEG2000 Encryption in JPSEC: Security, Applicability, and the Impact of Compression Parameters

Dominik Engel, Thomas Stütz, and Andreas Uhl

Department of Computer Sciences, Salzburg University, Jakob-Haringerstr. 2, 5020 Salzburg, Austria

Correspondence should be addressed to Andreas Uhl, uhl@cosy.sbg.ac.at

Received 30 March 2007; Revised 2 July 2007; Accepted 22 October 2007

Recommended by E. Magli

JPEG2000 encryption has become a widely discussed topic and quite a number of contributions have been made. However, little is known about JPEG2000 compression parameters and their influence on the security and performance of format, compliant encryption schemes. In this work, a thorough analysis of this topic is presented with a focus on format-compliant packet body encryption as sketched in the FCD 15444-8 (JPSEC). A proof for the reversibility of JPSEC format-compliant packet body encryption is given. As format-compliant packet body encryption preserves the JPEG2000 headers, which severely compromises the security, we additionally discuss packet header encryption with a special focus on format compliance and the influence of compression parameters on these schemes.

1. INTRODUCTION

The demand for custom encryption methods specifically tailored to fulfill the requirements of a particular multimedia application scenario with respect to security and other functionalities is widely accepted nowadays [1, 2].

Besides the aim to reduce the computational effort (which is usually achieved by trading off security as it is the case in partial/selective or soft encryption schemes), maintaining format compliance is the major goal of dedicated encryption techniques for visual media. In case of format compliance after encryption, the properties of the original media format file carry over to the encrypted file. For example, rate adaptation may be done in the encrypted domain easily, provided the original format supports this functionality as well, which is true for scalable or embedded bitstreams, for example.

Two types of media encryption technologies typically support format compliance to a certain extent: "bitstream-oriented" and "compression-integrated" methods. In the first case, encryption is applied directly to the bitstream and, in most cases, header information is left in plaintext. The actual visual information is encrypted avoiding the emulation of marker and header sequences in the ciphertext parts ("bitstream-compliant encryption" of visual data, in the case of JPEG2000 sequences in excess of `0xff8f` must not be generated as these are reserved for marker sequences). Overall, this approach leads to a format-compliant bitstream (in Section 1.2, the different forms of compliance are discussed more precisely and in more detail).

Leaving the headers in plaintext severely compromises the security, as we will show more in more detail in Sections 3.2 and 4.2. Therefore, in order to ensure full confidentially and minimize the information leakage, the packet headers have to be encrypted as well either in bitstream-compliant manner (i.e., encrypted headers do not generate marker or header sequences) or even in "format-compliant" manner (i.e., encrypted headers represent syntactically and to some extent semantically correct header information). Compression-integrated encryption on the other hand interleaves compression with encryption or simply integrates encryption into the compression pipeline which leads to an intrinsic support of format compliance for most approaches.

Both types of techniques have been discussed extensively in the context of JPEG2000 encryption. While limiting the impact of encryption on compression performance of the overall scheme is a crucial issue for compression-integrated techniques (see [3–5] for some examples in the JPEG2000 context), the potential impact of compression settings within JPEG2000 on the security of the corresponding bitstream-oriented encryption techniques has not been investigated so far. This is the aim of this paper.

In the remaining parts of the introduction, we shortly review the JPEG2000 compression pipeline and the data format of JPEG2000 bitstreams, and we describe the image quality metrics used in our assessments. Section 2 describes the format-compliant packet body encryption as sketched in the FCD 15444-8 [6] (which is now an officially published standard [7]) and proves its reversibility. In Section 3, we discuss the impact of compression settings in JPEG2000 on the security of packet body-based encryption techniques. On the one hand, this relates to information contained in JPEG2000 headers for different compression parameters (headers are left in plaintext in this approach which in turn leads to a certain amount of information leakage). On the other hand, the security of partial/selective encryption schemes is shown to be severely affected by different compression parameter settings. Section 4 discusses bitstream-compliant and format-compliant ways to encrypt header information to limit information leakage and to increase security. Again, the relevance of different compression parameter settings in this context is analyzed. Section 5 concludes the paper.

1.1. JPEG2000 compression pipeline and parameters

JPEG2000 [8] partitions the image into rectangular tiles; the default of the reference software (JasPer (http://www.ece.uvic.ca/~mdadams/jasper) and JJ2000 (http://jj2000.epfl.ch)) is only one tile for the entire image. On each tile, JPEG2000 employs a wavelet transform: either an irreversible 9/7 or a lossless 5/3 wavelet transform can be chosen. Additionally, the number of pyramidal wavelet decompositions can be set, which corresponds to the number of obtained resolutions minus one. The default value of the JPEG2000 reference software is five decomposition levels.

After the wavelet transform, the coefficients are quantized and encoded on a codeblock basis using the EBCOT scheme, which renders distortion 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 codeblock size can be chosen arbitrarily with certain restrictions. The codeblock width and height are limited to powers of two with the minimum size being 2^2 and the maximum being 2^{10} [9, page 32]. Furthermore, the number of coefficients contained in a codeblock must be less or equal to 4096. Hence the smallest codeblock is 4×4 and the largest square codeblock is 64×64, which is the default value of the reference software. The coding passes may contribute to a certain quality layer. The number of quality layers (and their rates) can be specified. While JJ2000 uses as many as 32 quality layers by default, only a single quality layer is employed by the Jasper coder. A packet body contains codeblock contribution to packets (CCPs) of codeblocks of a certain resolution, quality layer and precinct of a certain tile of a component. A precinct is a spatial intersubband partitioning structure that contains one to several codeblocks. The precinct size can be set as well, by default the reference coders only employ one precinct. A precinct enables spatially local processing, for example, cropping of an image in the compressed domain is possible by merely dropping JPEG2000 packets.

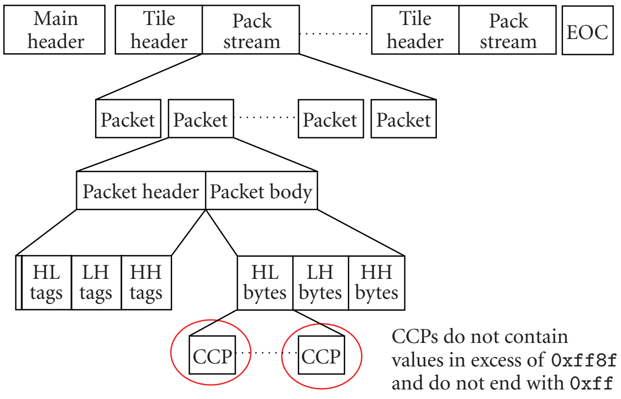


Figure 1: Restrictions within the CCPs.

The JPEG2000 codestream consists of headers (main header, tile headers, tile-part headers) and packets that consist of packet headers and packet bodies (cf., Figure 1). The compressed coefficient data is contained in the packet bodies. The packet header contains information about the CCPs of the codeblocks, for example, the number of coding passes and the CCP lengths (further details are discussed in Section 3.2). The packet header and the packet bodies must not contain any two-byte sequence in excess of `0xff8f` nor end with an `0xff` byte (bitstream compliance). The arithmetic coding of the bitplanes is referred to as tier1 encoding, while the partitioning of the coding passes into quality layers and the generation of the packet headers is referred to as tier2 encoding.

Depending on the progression order (specified for a tile), the packets are ordered by resolution, quality layer, precinct, and component. There are five progression orders defined as follows.

(i) Layer-resolution level-component-position progression.

(ii) Resolution level-layer-component-position progression.

(iii) Resolution level-position-component-layer progression.

(iv) Position-component-resolution level-layer progression.

(v) Component-position-resolution level-layer progression.

The most interesting ones are layer-resolution level-component-position progression (LRCP or simply layer progression) and resolution level-layer-component-position progression (RLCP or simply resolution progression), which contain different versions of the same image at different rates or resolutions.

1.2. Bitstream, format, and JPEG2000 compliance

The term “bitstream” in its original meaning refers to an arbitrary stream of bits, which is referred to in the term “bitstream-oriented”. However in the JPEG2000 standard, it has a precisely defined alternate meaning. According to the

JPEG2000 standard [9], the term "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." [9, page 2], Coded symbols do not contain sequences in excess of `0xff8f`, which are avoided by a bit stuffing routine in the arithmetic coder. These sequences are used to signal in bitstream marker segments and thus will not be generated in the encryption process (schemes fulfilling this requirement and avoiding `0xff` bytes at the end of a processing unit are denoted as bitstream-compliant). An encryption scheme delivering a valid JPEG2000 codestream is denoted as format-compliant. Part 4 of the JPEG2000 standard suite (conformance testing) [10] 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; any other requirements using quality criteria are not part of the standard [10, page 30]). JPEG2000 compression with a compliant encoder followed by any bitstream-compliant packet body encryption scheme is therefore JPEG2000 compliant in the sense of [10].

1.3. A note on image and security metrics

The peak signal-to-noise ratio (PSNR) is by far no optimal choice for assessing the level of distortion of low-quality images. A number of suggestions can be found for alternative measures, but there is not much work on assessing low-quality images,such as those that are typically obtained by attacks on encrypted visual data. Mao and Wu [11] propose a measure that separates evaluation of luminance and edge information into a *luminance similarity score* (LSS) and an *edge similarity score* (ESS), reflecting properties of the human visual system. According to the authors, this measure is well suited for assessing distortion of low quality images. LSS behaves in a way very similar to PSNR. ESS is the more interesting part in the context of the work presented here, as it reflects the extent for structural distortion. ESS is computed by block-based gradient comparison and ranges, with increasing similarity, between 0 and 1. We use the weights and block sizes proposed by [11] in combination with Sobel edge detection. In this paper, apart from PSNR results, ESS results will be given for the evaluations. However, ESS is not a reliable estimator of the subjectively perceived image quality either and therefore the presentation of visual examples is absolutely inevitable.

1.4. Format-compliant encryption in JPSEC

The JPSEC security framework provides security services for JPEG2000 compressed images [12, 13]. A wide range of security services can be implemented within the JPSEC standard, including authentication, verification, conditional access, and confidentiality. The standard mostly defines the syntax tools to communicate the actually employed security services, their parameters, and the parts of the JPEG2000 file on which they are employed (Zone of Influence, ZOI). The standard also contains an informative section on technology examples, which outlines the intended usage of the standard framework. In the context of this paper, we focus on encryption. Format-compliant encryption schemes can easily be implemented in JPSEC by employing the so-called user-defined tools. These enable the usage of format-compliant encryption tools, a reference is the informative part of technology examples in [14, page 97], that specifies a packet body-based approach. Selective/partial encryption schemes are supported by the appropriate definition of ZOI. Packet-header encryption is therefore simply applying an encryption tool to the corresponding ZOI, namely, the packet-header portion of the codestream.

2. JPSEC PACKET BODY ENCRYPTION

The encryption of the JPEG2000 packet body data has been proposed in quite a number of contributions [12, 15–23]. If the packet body encryption preserves bitstream compliance (no sequence in excess of `0xff8f` nor an `0xff` at the end) and does not alter the length of the packet body data (e.g., by stuffing bits to avoid marker sequences), the encrypted JPEG2000 codestream is format-compliant and the packet headers can be left unchanged.

In the Annex B.5 of the JPSEC standard [6, 7], a method for format-compliant encryption is sketched, which encrypts the packet bodies and allows selective and full encryption of JPEG2000 codestreams. The document, however, does not contain all necessary details to implement the method. On the contrary, it is noted that the proof of the reversibility of the scheme is still missing. The packet body is split into two-byte sequences. The encryption process is defined in the following way.

> Every two-byte sequence of the packet body bytes 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 output as ciphertext.

If the length of the plaintext is odd, it is proposed to leave the last 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.

> 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 output as plaintext.

It is notable that the underlying encryption routine for two byte sequences must satisfy the following property: $d(x) = e(x)$ and thus $e(e(x)) = x$. This is met by all encryption modes that apply XOR on the plaintext with a key stream (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 may possibly be generated).

We have found an interpretation for the term "relating code" which makes the scheme reversible (a proof is given in Section 2.4).

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 two-byte 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 temporarily encrypted sequence I_j is outputted as ciphertext (else, the plaintext P_j is outputted).

(E1) $I_j \leq \texttt{0xff8f}$
Necessary to obtain a bitstream-compliant two-byte ciphertext sequence.

(E2) $P_{j-1} \mid I_j \leq \texttt{0xff8f}$
Necessary to ensure bitstream compliance if the previous two-byte sequence has been left in plaintext.

(E3) $I_{j-1} \mid I_j \leq \texttt{0xff8f}$
Necessary to ensure bitstream compliance if the previous two-byte sequence has been replaced by the temporarily encrypted sequence.

(E4) $I_j \mid P_{j+1} \leq \texttt{0xff8f}$
Necessary to be able to preserve the next two-byte sequence in plaintext.

(E5) $I_{j-1} \mid P_j \leq \texttt{0xff8f}$
Necessary to detect (E4) for $j - 1$.

If the following conditions are met, then the temporarily decrypted sequence D_j is outputted as plaintext (else the ciphertext C_j is outputted).

(D1) $D_j \leq \texttt{0xff8f}$
Detection of the violation of (E1) (if (E1) has not been met, (D1) is not met and the ciphertext is the plaintext).

(D2) $P_{j-1} \mid D_j \leq \texttt{0xff8f}$
Detection of the violation of (E2).

(D3) $I_{j-1} \mid D_j \leq \texttt{0xff8f}$
Detection of the violation of (E3).

(D4) $D_j \mid C_{j+1} \leq \texttt{0xff8f}$
Detection of the violation of (E4).

(D5) $I_{j-1} \mid C_j \leq \texttt{0xff8f}$
Detection of the violation of (E5).

All conditions referencing undefined bytes (e.g., P_{-1}) are by default true. Note that the last byte requires special treatment. The best solution (in terms of maximum encryption percentage) is to modify (E1) and (D1) such that an `0xff` at the end is forbidden. Single-byte packets and the last byte of an odd-length packet may be encrypted modulo `0xff`.

2.1. Compression

There is no influence on compression performance.

2.2. Security

The headers are preserved. Information leakage occurs whenever a two-byte sequence of plaintext is preserved. Our experiments reveal that about every 128th byte is preserved. However, the preserved two byte sequences are not distinguishable from the encrypted sequences (compared to the other bitstream-compliant approaches [12, 18, 20, 21] that always preserve clearly distinguishable parts of plaintext).

2.3. Performance

There is a slight performance overhead compared to conventional encryption (e.g., AES in OFB mode) due to the additional comparisons (five conditions for every two-byte sequence). However, the approach is extremely well performing compared to other packet body-based encryption approaches.

2.4. Proof of reversibility

We first prove the reversibility of the algorithm for even-length packet bodies, the odd-length packet bodies simply follow. The proof is structured in the following way. First we show the reversibility for packet bodies with only two bytes and then for packet bodies consisting of n two-byte sequences. For a packet consisting of n two-byte sequences, we show the reversibility for the first-two byte sequence, then for the jth two-byte sequence (give that the $(j-1)$th sequence has been correctly decrypted) and finally for the last two-byte sequence.

2.4.1. Reversibility for packets of length 2

For the case of a single two-byte sequence, only the first encryption and decryption rule apply (extended with the requirement that the last byte must not equal `0xff`). There are two cases either $C_1 = P_1$ or $C_1 = I_1$. If $C_1 = P_1$, then $I_1 = e(P_1)$ cannot be bitstream-compliant and therefore the same holds for $D_1 = d(P_1) = e(P_1)$. If $C_1 = I_1$, then $I_1 = e(P_1)$ is bitstream-compliant and therefore the same holds for $D_1 = d(P_1) = e(P_1)$.

2.4.2. Reversibility for the first two-byte sequence for packets of length $2n$

For the first two byte sequence there are again two cases.

If $C_1 = P_1$, either (E1) or (E4) has not been met since the other rules do not apply.

> If (E1) is not met, then (D1) is not met (same argument as before).
>
> If (E4) ($I_1 \mid P_2 \leq$ 0xff8f) is not met for the first sequence, (E5) ($I_1 \mid P_2 \leq$ 0xff8f) is not met for the second sequence and $C_2 = P_2$. Therefore (D4) ($D_1 \mid C_2 = I_1 \mid P_2 \leq$ 0xff8f) is not met.

If $C_1 = I_1$, then (E1) and (E4) have been met and we have to show that all of the decryption rules are met. Again, only (D1) and (D4) apply.

> (D1) is met because $d(I_1) = e(I_1) = P_1$ which is bitstream-compliant by precondition (it is a packet body sequence).

(D4) is met because $D_1 \mid C_2 = P_1 \mid C_2$, which is either $P_1 \mid P_2$ (which is less than or equal to `0xff8f` due to the precondition of bitstream-compliance) or $P_1 \mid I_2$. C_2 is only set to I_2, if (E5) ($P_1 \mid I_2 \leq$ 0xff8f) is met.

As both (D1) and (D4) are met, the first cipher sequence is correctly decoded.

2.4.3. *Reversibility for the jth two-byte sequence for packets of length $2n$*

In the following, we show that we can decode the jth cipher sequence ($1 < j < n$) correctly, given that we have decoded the $(j-1)$th sequence correctly.

If $C_j = P_j$, we have to show that for any violation of an encryption rule, a decryption rule is not met either.

If (E1) is not met, it is obvious that (D1) is not met.
If (E2) ($P_{j-1} \mid I_j$) is not met, then (D2) ($P_{j-1} \mid D_j = P_{j-1} \mid I_j$) is not met either.
If (E3) ($I_{j-1} \mid I_j$) is not met, then (D3) ($I_{j-1} \mid D_j = I_{j-1} \mid I_j$) is not met either.
If (E4) ($I_j \mid P_{j+1}$) is not met, then (E5) for $(j+1)$ ($I_{(j+1)-1} \mid P_{(j+1)}$) is not met either and $C_{j+1} = P_{j+1}$. Therefore (D4) ($D_j \mid C_{j+1} = I_j \mid P_{j+1}$) is not met.
If (E5) ($I_{j-1} \mid P_j$) is not met, then (D5) ($I_{j-1} \mid C_j = I_{j-1} \mid P_j$) is not met either.

If $C_j = I_j$, we have to show that if all encryption rules are met, it follows that all decryption rules are met.

(D1) is met because (E1) is met.
(D2) ($P_{j-1} \mid D_j = P_{j-1} \mid P_j$) is met because $P_{j-1} \mid P_j$ is preconditionally codestream-compliant.
(D3) ($I_{j-1} \mid D_j = I_{j-1} \mid P_j$) is met because of (E5).
For (D4), we have to look at $D_j \mid C_{j+1} = P_j \mid C_{j+1}$, where C_{j+1} is either P_{j+1} or I_{j+1}. $P_j \mid P_{j+1}$ is trivially met (precondition). C_{j+1} is only set to I_{j+1} if (E2) ($P_{(j+1)-1} \mid I_{(j+1)}$) is met for $(j+1)$. Hence (D4) is met.
(D5) $I_{j-1} \mid C_j = I_{j-1} \mid I_j$ is met because (E3) is met.

Thus we can decode C_j correctly.

2.4.4. *Reversibility of the last two-byte sequence for packets of length $2n$*

For the nth sequence, there are again two cases ($C_n = P_n$ or $C_n = I_n$).

If $C_n = P_n$, then we have to show that for any encryption rule which is not met, a decryption rule is not met either.

If I_n is in excess of `0xff8f` or ends with `0xff`, then obviously the same holds for D_n, which is $e(C_n) = e(P_n) = I_n$.
If (E2) ($P_{n-1} \mid I_n$) is not met, then (D2) ($P_{n-1} \mid D_n = P_{n-1} \mid I_n$) is not met either.
If (E3) ($I_{n-1} \mid I_n$) is not met, then (D3) ($I_{n-1} \mid D_n = I_{n-1} \mid I_n$) is not met either.
(E4) does not apply.
If (E5) ($I_{n-1} \mid P_n$) is not met, then (D5) ($I_{n-1} \mid C_n = I_{n-1} \mid P_n$) is not met

If $C_n = I_n$, then we have to show that if all encryption rules are met, it follows that all decryption rules are met.

If I_n is not in excess of `0xff8f` nor ends with `0xff`, then the same holds for D_n, which is $e(C_n) = e(I_n) = P_n$.
(D2) ($P_{n-1} \mid D_n = P_{n-1} \mid P_n$) is met because $P_{n-1} \mid P_n$ is preconditionally bitstream-compliant.
(D3) ($I_{j-1} \mid D_j = I_{j-1} \mid P_j$) is met because of (E5).
(D4) does not apply.
(D5) ($I_{n-1} \mid C_n = I_{n-1} \mid I_n$) is met because (E3) is met.

Putting everything together, we have proven that the scheme is reversible for packet bodies consisting of arbitrary n two-byte sequences. The odd cases simply follow if we preserve the last byte. Then according to the proof, all sequences except the last single byte are reversibly decodable, and the last byte is simply preserved (trivially reversible).

3. COMPRESSION PARAMETERS AND PACKET BODY-BASED ENCRYPTION

Basically, compression parameters influence the security of the packet body-based encryption approaches, such as the JPSEC proposal, in different ways. If the entire packet body data is encrypted, the only information preserved is the headers.

It is commonly assumed that packet body-based encryption approaches are capable of offering a high level of security. Common terms to describe the capability of a packet body-based encryption scheme are “full encryption” [7, page 87], “full protection” [22], “total encryption” [24]; in [21, 23] “efficient and secure encryption schemes” are proposed and the selective encryption approach of [16] offers a “high level of confidentiality”. The influence of compression parameters on the security of a packet body-based scheme is not discussed in literature and therefore we address this issue here in Section 3.2. The entire Section 3.2 is dedicated to the analysis of the information contained in the JPEG2000 headers for different compression parameters. Section 4 discusses possible approaches for encrypting this information.

If only a fraction of the packet body data is encrypted, the compression parameters are the major influence on the security of these schemes. Norcen and Uhl have shown in [16] that the encryption of the leading 20% of the JPEG2000 codestream is sufficient to ensure confidential encryption for certain coding settings. In Section 3.3.2, their approach is analyzed and generalized for other compression parameter settings.

In fact, the compression parameters also have an enormous influence on the efficiency of packet body-based selective/partial encryption approaches.

3.1. *Performance and compression parameters*

JPEG2000 offers features for improved error resilience, for example, the optional insertion of an SOP marker segment to signal the start of a packet header and the optional insertion of EPH markers which signals the end of a packet header. These markers enable easy and efficient transcodability, because the identification of the packets is an extremely simple parsing process for the two-byte marker sequences SOP (`0xff91`) and EPH (`0xff92`). Norcen and Uhl [16] were the first to propose the exploitation of this mechanism for selective encryption, namely, for the identification of the packet body borders. Thereby, the complexity of format-compliant packet body-based encryption approaches is greatly reduced and the efficient transcodability is preserved in the encrypted domain. Hence a computationally weak network node is capable of transcoding even encrypted high-definition JPEG2000 streams efficiently.

3.1.1. *Influence on compression performance*

The usage of the SOP and EPH marker segments has a certain negative influence on compression performance if many and relatively short packets are produced. If SOP and EPH marker segments are used, every packet needs an additional 8 bytes. The SOP marker segment takes up 6 bytes, namely, the two-byte SOP marker, followed by two bytes (Lsop) indicating the length of the packet header and a two-byte packet index (Nsop). The EPH marker is solely the two-byte marker code. The number of quality layers q, the number of decomposition levels d, and the number of precincts p each linearly increase the number of packets for a specific tile of a component. More precisely, the number of packets n for a specific tile of a component is given by $n = q(d+1)p$. However, a sensible choice of the parameters q, d, and p keeps the induced overhead moderate. For JJ2000 default settings (32 quality layers, 5 decomposition levels, 1 precinct), we compress the well-known Lena image with 512 × 512 pixel at 2.79 bpp.The usage of SOP and EPH marker segments reduces the compression performance to 2.84 bpp, which results in a moderate overhead of 1.68% (the compressed size with markers is 101.68% of the compressed size without markers). For Jasper default settings (1 quality layer, 5 decomposition levels, 1 precinct), the overhead is negligible (0.053%).

Certain format-compliant JPEG2000 encryption approaches need special compression parameters in order to perform efficiently. Stütz and Uhl [25] have shown that the usage of enough quality layers is, for example, crucial for the iterative encryption approach as proposed by Wu and Deng [22].

3.2. *Information within 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, …). This information is mostly rather generic and does not reveal much more than the fact that the attacker has to deal with encrypted JPEG2000 files. Nevertheless, it has to be kept in mind that the image size, the number of components, and the compression parameters can be accessed in plaintext. This information will in general be insufficient to map a specific image to its encrypted version.

Further, the packet headers have to be taken into account. The packet header contains information about lengths of the CCP (their sum is the length of the packet body), the number of coding passes, inclusion information (in which quality layer is the codeblock which is included first), and the number of leading zero bitplanes (lzb). In the tier1 encoding process, the codeblocks are encoded bitplane per bitplane except for the leading zero bitplanes. The number of leading zero bitplanes depends on the source image and will give more precise information on the actual coefficient value if the codeblocks are smaller. The sign bit is coded independently and therefore the leading zero bitplanes only give information about the absolute coefficient values. If no wavelet transform is applied, the information of the leading zero bitplanes leads to a crude quantization on a block basis (cf. Figure 2). Hence large codeblocks (as the number of coefficients has to be below 4096, the largest square codeblock is 64 × 64) are absolutely necessary to hide the image content if the packet headers are preserved. If more wavelet decomposition levels are applied, the reconstructions seem to contain no image information at all (cf. Figure 3). However, this conclusion is wrong, as Figure 4 reveals. The LH1, HL1, and HH1 subbands are not further processed in the pyramidal wavelet decomposition and hence these subbands can be reconstructed and their image information is recoverable. Therefore, large codeblocks have to be employed for wavelet decompositions of more levels as well, in order to hide the image content.

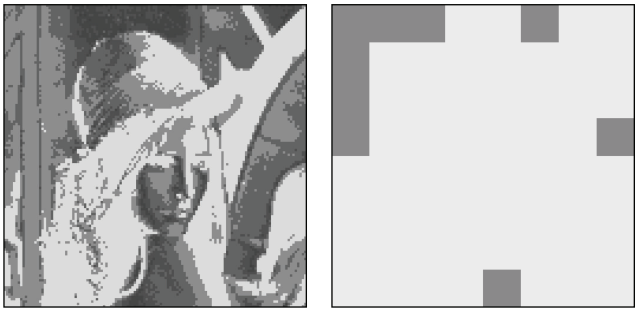
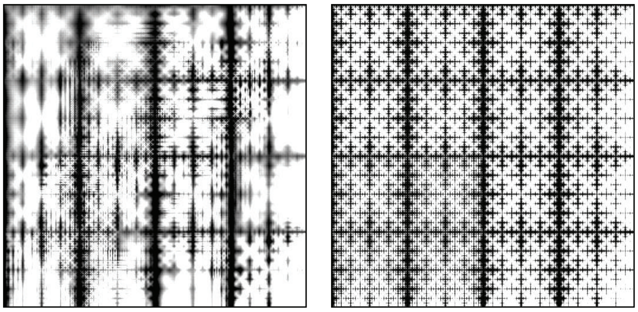

(a) 4 × 4 codeblocks (b) 64 × 64 codeblocks

Figure 2: No wavelet transform: reconstruction based on lzb information.

If high resolution visual data is encrypted, the lzb information may severely leak image information even for larger codeblock sizes (up to the maximum codeblock size). As JPEG2000 is employed in the digital cinema specification [26], we present a visual example with the maximum vertical resolution of the 4 k case (2160 pixel). In Figure 5, the original image and the reconstruction based on the lzb information with respect to a codeblock size of 16 × 16 are shown. However, the image content is perceivable in the lzb reconstruction with respect to a codeblock size of 32 × 32, and even for the maximum codeblock size, the image content is discernible (see Figure 6).

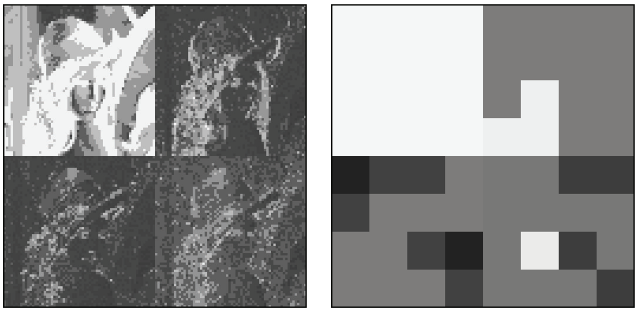
In the tier2 encoding process, the coding passes of a codeblock are assigned to a certain quality layer. Even for fixed compression parameters, the results will vary depending on the given image. Hence the CCP lengths (and therefore the packet body length) and the inclusion information will differ



(a) 4 × 4 codeblocks

(b) 64 × 64 codeblocks

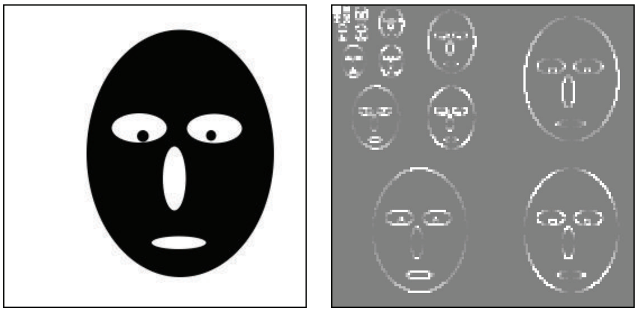
Figure 3: Reconstruction on basis of the lzbs after a wavelet transform with 7 pyramidal decomposition levels.



(a) 4 × 4 codeblocks

(b) 64 × 64 codeblocks

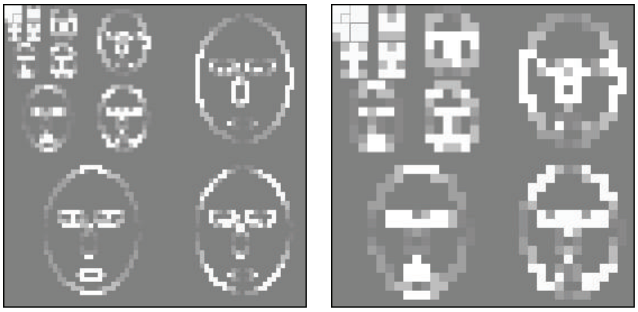
Figure 4: The lzb information for the subbands of a 1-level wavelet transform.



(a) Original, resolution 2160 × 2160

(b) 16 × 16 codeblocks

Figure 5: The lzb information of a high-resolution image



(a) 32 × 32 codeblocks

(b) 64 × 64 codeblocks

Figure 6: The lzb information of a high-resolution image

from image to image. In this way, the header information reveals a rather distinctive fingerprint of an image, enabling an attacker to link a specific image to an encrypted one. The distinctiveness of the contained fingerprint is experimentally shown in Section 4.2. Several proposals that employ a fingerprint which uses the details of the packet headers [27–29] can be found in literature. However, deducting the image content from the inclusion and CCP lengths information does not seem to be possible if the coefficient data is securely encrypted.

Many quality layers and a smaller codeblock size increase the amount of information contained in the packet headers, for example, there are 440 bytes of nonpacket body bytes opposed to 140547 packet body bytes for only a single quality layer and 64 × 64 codeblocks compared to 10242 bytes and 94918 bytes for 16 × 16 codeblocks and 32 quality layers for the Lena image (512 × 512). About 150 bytes are main and tile-part headers in both cases. For every packet, 8 bytes are used for the SOP marker segment and EPH marker, which do not contain additional information. This means that there are only about 300 packet header bytes for the first case and about 8600 for the second case.

Hence large codeblocks (absolutely inevitable) and few quality layers improve the security of format-compliant packet body-based encryption approaches such as the JPSEC proposal. For high-resolution images, even the maximum codeblock size can leave discernible image information in the packet headers. If confidentiality and thus the reduction of information leakage is of importance, the packet headers have to be encrypted as well, as described in Section 4. If header encryption is used, then it is no longer necessary to restrict coding to few quality layers and large codeblocks.

3.3. *Zone of influence/partial encryption*

The encryption of only a fraction of the actual data has basically two motivations: first, a reduction of complexity is of great benefit; and second, more sophisticated application scenarios (transparent encryption) can be implemented. The encryption of massive amounts of image data is still an enormous task. An example is that the encryption of high-definition video (JPEG2000 is employed as frame compression codec in the DCI's digital cinema specification [26]) is still an enormous computational task. An approach for confidential encryption with reduced complexity [16] and the influence of JPEG2000 compression parameters is discussed in Section 3.3.2.

Other application scenarios do not strive for maximum security but impose additional requirements. Macq and Quisquater [30] introduce the term "transparent encryption" in the context of digital TV broadcasting. A broadcaster's goal is to attract viewers in order to increase the number of paying customers. Hence a low-quality version is publicly available, while the full-quality version is intended for paying customers only. There are two major requirements that have to be met concurrently: hiding a specific amount of image information (security requirement) and showing a specific amount of image information (quality requirement). JPEG2000 format-compliant encryption can be employed to implement transparent encryption [31]; the desired low-quality version is left in plaintext and the enhancement information is encrypted, thereby taking advantage of the scalability of the JPEG2000 codestream. In Section 3.3.3, transparent JPEG2000 encryption and the influence of compression parameters are discussed.

If only parts of the data are encrypted, the direct reconstruction of the format-compliantly encrypted image (which is possible with every standard-compliant JPEG2000 decoder) is not the best reconstruction an attacker may be able to achieve.

3.3.1. Attacks

The encrypted parts introduce severe noise, but an attacker may identify the encrypted parts and replace them, thereby enhancing the image quality.

The *concealment attack* has been proposed in [16] and employs the JPEG2000 built-in error concealment mechanisms. At the end of each bitplane (cleanup pass), an additional symbol (`0xa`, exactly the 4 bits 1001) is encoded; if it is not decoded correctly, the coding pass and all successive ones are discarded. Given that encryption results in an approximately uniformly distributed ciphertext (packet bodies), we can assume that the segmentation symbol is generated randomly in 1 out of 16 cleanup passes (thereby suggesting that this data is correct) and thus noise is introduced due to the uncorrected encrypted data.

The segmentation symbol is invoked by the `-Cseg_symbol on` option of the JJ2000 encoder. In this way, the actions of a possible attacker are mimicked.

In this context, it is notable that the reference JPEG2000 software JJ2000, which is actually the only software that offers error concealment at the decoder (concerning jasper, kakadu, and JJ2000), has a minor bug in the actual error concealment function (in the jj2000.j2k.entropy.decoder .StdEntropyDecoder in line 2475 (4.1 unix release) it should be "resetmask = $(-1) \ll (bp+1)$;" instead of "resetmask = $(-1) \ll (bp)$;"), which renders the error concealment mostly useless, as the erroneous bitplane is written and then the decoding of the codeblock is stopped.

If packet body encryption starts from the beginning of the codestream, it is best to set the coefficient to zero, regardless to which bitplane an error has been detected. In the case of contiguous encryption from the start, the coding passes are encrypted consecutively. Hence if an error is detected in a certain coding pass, all previous coding passes have been encrypted either.

A further improvement of this attack is the additional employment of the predictive termination of every coding pass (`-Cterm_type predict` and `-Cterminate on`), which leaves error-resilient information on the spare least significant bits, which can be used by a decoder to detect errors. However, our experiments revealed that the JJ2000 decoder is not capable of achieving a higher-detection rate with both error-resilience methods enabled, on the contrary, error resilience on the basis of the segmentation symbol performed best. This is due to a minor bug in the cleanup pass code of the JJ2000 decoder (in the jj2000.j2k.entropy.decoder.StdEntropyDecoder in line 2439 (4.1 unix release) it should be "error = error| mq.checkPredTerm();").

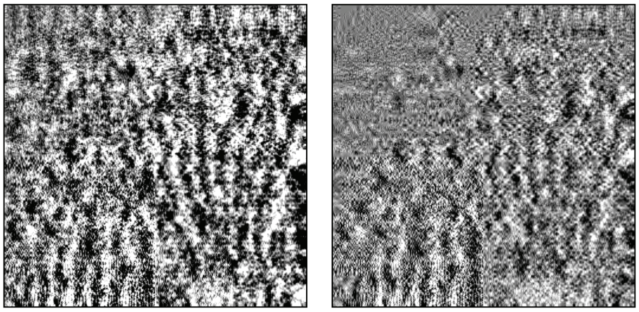
The decoder's behavior if errors occur is not standardized, only the detection mechanism is. A conservative strategy is to reset the coefficient value to the value prior to the error.

Another attack, especially useful against transparent encryption schemes, is the *truncation attack*. The truncation attack consists of simply truncating the encrypted parts of the codestream; in the case of the transparent JPEG2000 encryption approach [31], this is basically the best recoverable image quality, disregarding standard image enhancement processing and the information contained in the packet headers (cf. Section 3.2). It is easily applicable and does not require any specific encoding parameters. The start of encryption, however, has to be known or guessed (on the basis of the varying image quality). The JJ2000 decoder can be employed to conduct the attack with the following options: `-parsing off -nbytes`.

3.3.2. Confidential encryption

In [16], Norcen and Uhl evaluate a selective encryption approach which encrypts the leading packet body bytes. They analyze the influence of the progression order, namely, resolution and layer progression, on their proposed scheme. Apart from the progression order and the insertion of SOP and EPH marker segments, the JPEG2000 compression parameters have been set to JJ2000 default values. The images were either losslessly compressed or lossy at fixed compression ratio of 4 (2 bpp). The PSNR plots for varying encryption percentages are given (in percent of the codestream), both for a direct reconstruction and the concealment attack. For the lossless compression, layer progression reveals less image information and is therefore better suited for the approach, but for lossy compression, the results are ambiguous and far less significant. Norcen and Uhl [16] conclude that independent of the progression order, the encryption of the leading 20% of the packet body data is sufficient to hide all significant image information. Figure 7 illustrates the direct reconstruction and the corresponding erroneous concealment attack (without the bug fixed) if the leading 20% of the packet body data are encrypted.

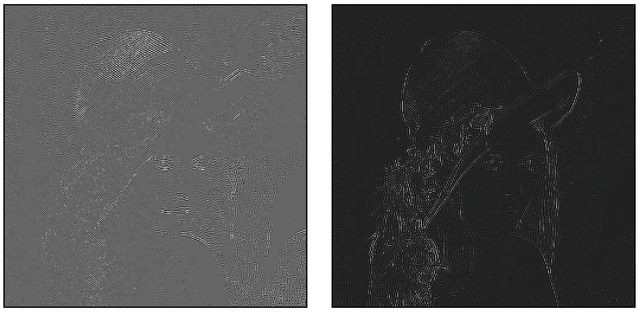
However, with the correct error concealment (the coefficients have been set to zero), this rule of thumb does not hold as Figure 8 reveals. As image quality metrics are no reliable estimators of the preserved image information for such low quality and sparse information, visual examples are inevitable. The actually necessary percentage to confidentially remove all image information contained in the packet body greatly depends on the source image and the compression parameters. No encryption percentage can be given, but resolution progression generally requires a higher-encryption percentage compared with layer progression (if enough quality layers are employed). If there are many quality layers, the codeblocks contribute to many different packets. In this way, the encryption of a packet affects the decoding of several other packets.



(a) Encryption: PSNR 8.4 dB, ESS 0.23

(b) Erroneous attack: PSNR 9.7 dB, ESS 0.23

Figure 7: Confidentiality with JPEG2000: 20% encrypted.



(a) Layer prog.: PSNR 14.50 dB, ESS 0.0

(b) Res. prog.: PSNR 14.53 dB, ESS 0.0

Figure 8: Concealment attack: 20% encrypted, 2 bpp.

The compression ratio has an enormous influence. In general it can be said that the higher the compression ratio, the higher the necessary encryption percentage.

Considering all possible source images and compression parameters, the only way to confidentially always hide all image information is to encrypt all of the packet body data. For example, Figure 9 shows the encryption of 80% from the start for 0.5 bpp and resolution progression: some high-frequency information is still visible.

Note that PSNR and ESS values correspond to the original reconstructions, the images in Figures 8 and 9 have been adjusted to improve the readability.

3.3.3. *Transparent encryption*

Transparent encryption can be implemented on top of scalable codestreams such as the JPEG2000 codestream (as proposed in [31]). Their relatively straightforward approach is to start encryption from a certain position to the end of the codestream. In [31], this approach is evaluated for several starting positions and for different progression orders, namely, layer and resolution progression. Both the direct reconstruction and the concealment attack have been conducted.

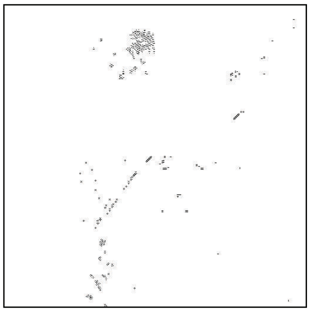


Figure 9: Concealment attack: 80% encrypted, 0.5 bpp, res. prog.: PSNR 14.50 dB, ESS 0.0.

For transparent encryption, it is desired that the gap between a direct reconstruction (as conducted by a potential costumer) and reconstruction of an attacker remains minimal. If the quality that an attacker can achieve is too good, no one will be interested in buying the full quality version. On the other hand, if the quality of the direct reconstruction is too low, the goal to attract customers will not be achieved. Ideally, the quality of a direct reconstruction is the same as the quality of the best attack.

In the following, we present several evaluations that show the influence of two compression parameters: progression order and codeblock size. Different error concealment strategies have been evaluated, namely, the usage of the segmentation symbol, predictive termination after each coding pass, and both methods combined. If both methods are combined and the coefficient values are reset on a coding pass basis (the value before the erroneous coding pass is exactly restored, while JJ2000's default is to reset them on a bitplane basis), the differences between the truncation attack and the error concealment attack are negligible (strictly below 1 dB and mostly zero). Hence for the sake of clarity, we only present the truncation attack in the plots.

Additionally, SOP and EPH markers are employed. The remaining compression parameters have been set to JJ2000 default settings with no target bitrate specified (nearly lossless). A test set of 100 images selected from the publicly available HDTV samples of the VQEG has been established. Three different resolution sets (each with the same 100 images) have been generated by downsampling the high-resolution images (3840 × 2160) to the resolutions 1024 × 576, 512 × 288 and 256 × 144.

Note that the usage of quality layers is necessary for these experiments to ensure that the leading contributions of the codestream already contain a version of the image of reasonable quality.

Interestingly, our experiments revealed that the codeblock size is the major factor for the success of the transparent encryption of JPEG2000. The smaller the codeblock size is chosen, the smaller becomes the gap between attack and direct reconstruction, and hence better suited for transparent encryption. This holds for all resolutions, as well as for layer (see Figures 10 and 11 for resolution 1024 × 576, Figures 14 and 15 for resolution 512 × 288, and Figures 18 and 19 for resolution 256 × 144) and resolution progressions

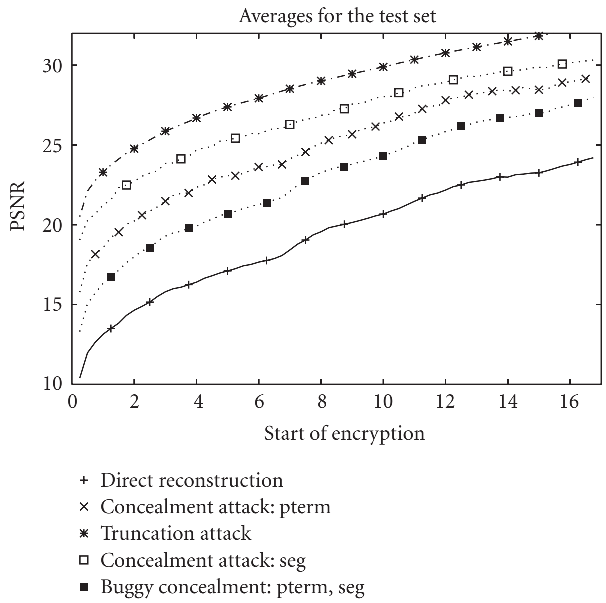


Figure 10: Average results for resolution 1024 × 576 and layer progression with default cblk size.

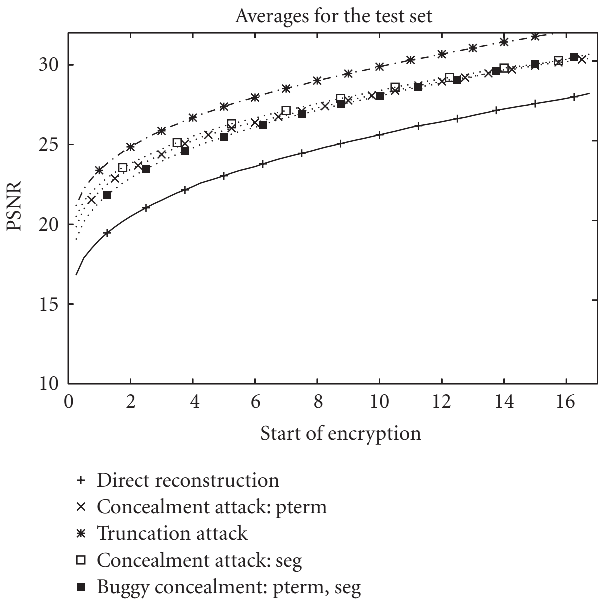


Figure 11: Average results for resolution 1024 × 576 and layer progression with a cblk size of 8 × 8.

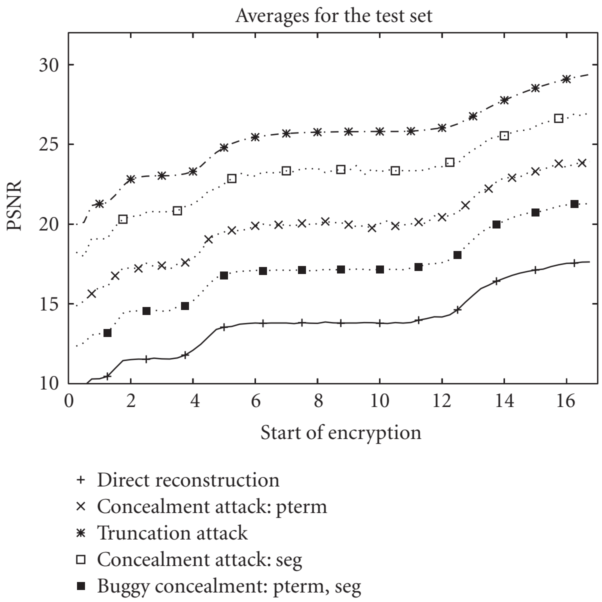


Figure 12: Average results for resolution 1024 × 576 and resolution progression with default cblk size.

(see Figures 12 and 13 for resolution 1024 × 576, Figures 16 and 17 for resolution 512 × 288, and Figures 20 and 21 for the resolution 256 × 144). These figures show PSNR plots for two different codeblock sizes (the maximum square codeblock size 64 × 64 and the very small codeblock size of 8 × 8) and for varying starts of encryption (given in percent of the codestream; the trailing packet body data is encrypted applying the JPSEC encryption approach). The PSNR values of the direct reconstructions, the concealment attacks, and the truncation attacks are plotted. These figures reveal that the truncation attack (which leads to almost the same results as the combined error concealment on a coding pass basis, that is therefore not plotted in the figures) always leads to higher PSNR values (better quality) than the single error concealment strategies, which lead to higher PSNR values compared to the direct reconstruction and the buggy error concealment (see Section 3.3.1).

Why do smaller codeblocks decrease the gap between direct reconstruction and reconstruction based on conducting an attack? As discussed in Section 3.2, the packet headers contain information about the leading zero bitplanes of a codeblock. Thus the smaller the codeblocks, the more accurate this information. The JPEG2000 decoder uses the lzb information to reconstruct the coefficients and hence the more accurate this information is, the more accurate becomes the reconstruction of the encrypted coefficients. Only the truncation attack does not decode any encrypted coefficient data at all and therefore no additional noise is introduced.

A smaller codeblock size decreases the compression efficiency and codeblock sizes below 8 × 8 cannot be recommended. The progression order mainly influences the increase of image quality in relation to the contributed codestream data. While the image quality increases smoothly if layer progression is employed, the image quality steeply increases at the beginning of each new resolution level (in the codestream data) and increases only very slightly at the end of a resolution for resolution progression. The progression order mainly influences the start of encryption for a desired image quality. However, the reduction of complexity remains small compared to the fact that over 95% of the JPEG2000 file have to be encrypted in order to severely reduce the quality. Visual examples are given for layer progression (see Figures 23 and 22) and resolution progression (see Figures 25 and 24).

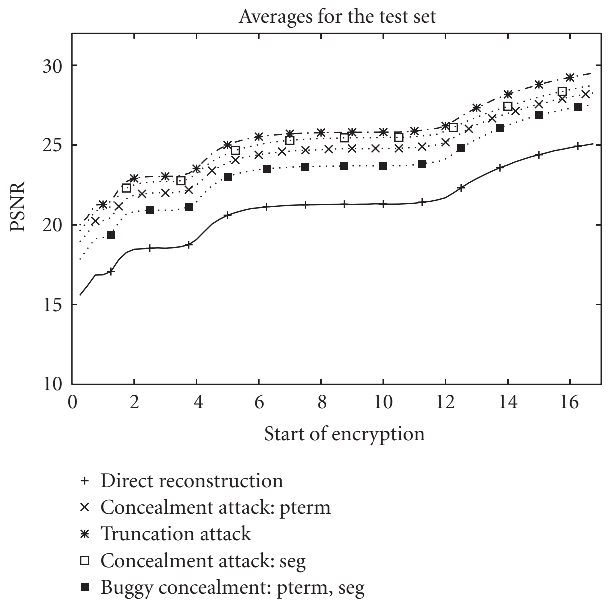


FIGURE 13: Average results for resolution 1024 × 576 and resolution progression with a cblk size of 8 × 8.

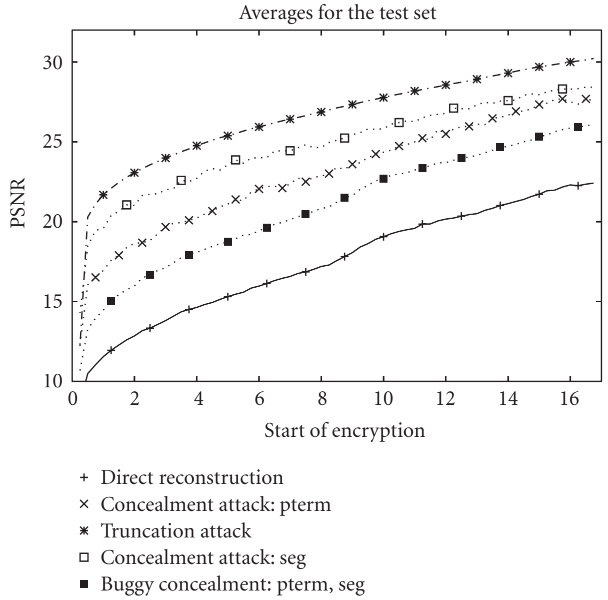


FIGURE 14: Average results for resolution 512 × 288 and layer progression with default cblk size.

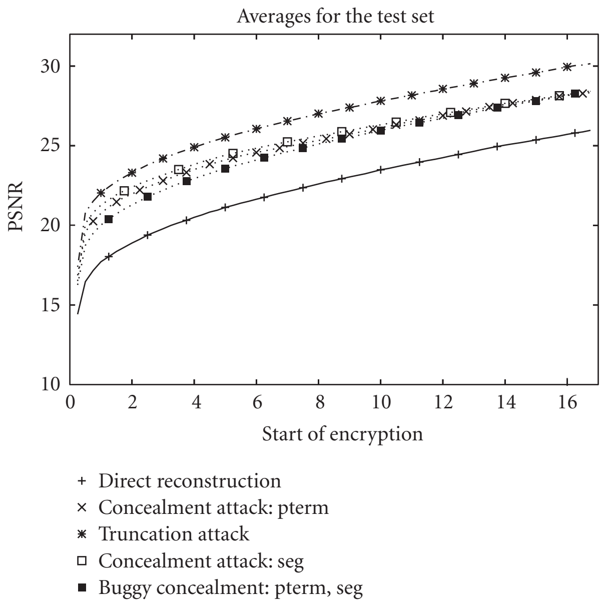


FIGURE 15: Average results for resolution 512 × 288 and layer progression with a cblk size of 8 × 8.

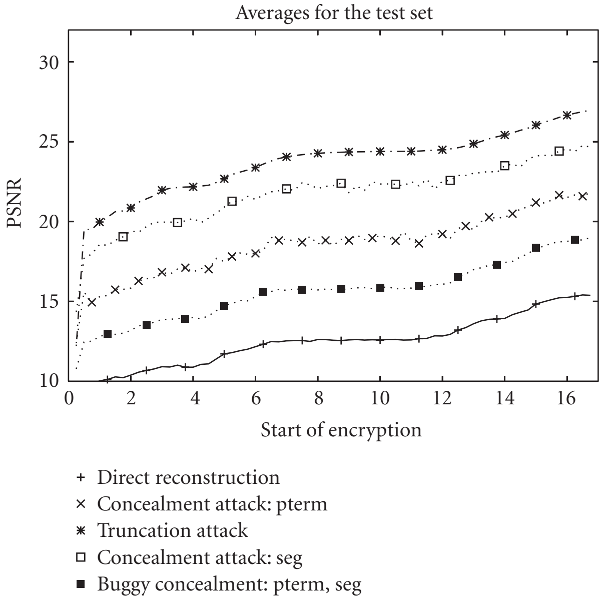


FIGURE 16: Average results for resolution 512 × 288 and resolution progression with default cblk size.

4. HIGH-LEVEL SECURITY AND PACKET HEADER ENCRYPTION

As discussed in Section 3.2, the packet headers contain a fingerprint and even content-related data (leading zero bitplane information) of the image. In order to increase the security, the packet headers can be encrypted, while preserving the packet-based scalability. This can either be done by bitstream-compliantly encrypting both packet headers and packet bodies or by format-compliantly encrypting the packet headers.

4.1. Proposal of a bitstream-compliant encryption scheme

If SOP and EPH markers are applied, the bitstream-compliant encryption (with the JPSEC approach) of the packet headers and packet bodies produces an encrypted file

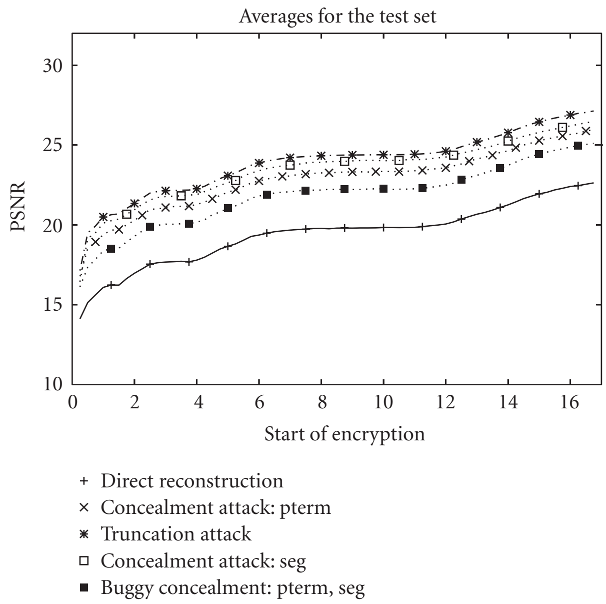


Figure 17: Average results for resolution 512 × 288 and resolution progression with a cblk size of 8 × 8.

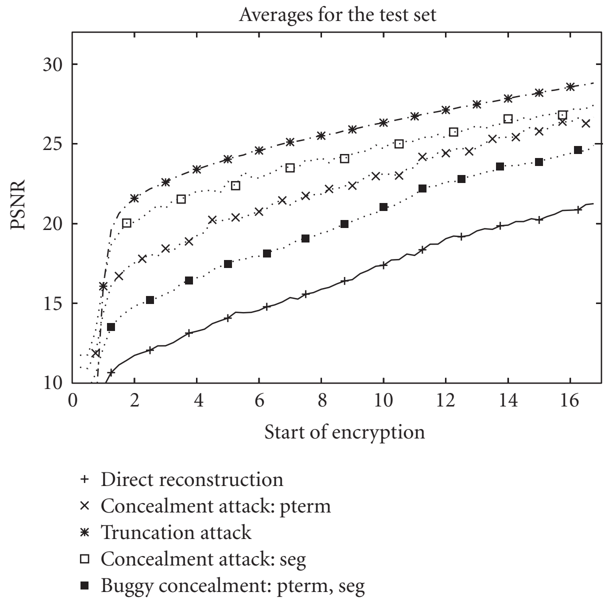


Figure 18: Average results for resolution 256 × 144 and layer progression with default cblk size.

that is format-compliant on a lower level (before tier2 decoding). Thus rate adaption by packet dropping can be applied in the same way as for a JPEG2000 file. Compression performance is not influenced at all. The JPSEC approach preserves on average every 128th byte, which will result in a slightly higher correlation between plaintext and ciphertext as compared to full encryption with a traditional cipher, such as AES.

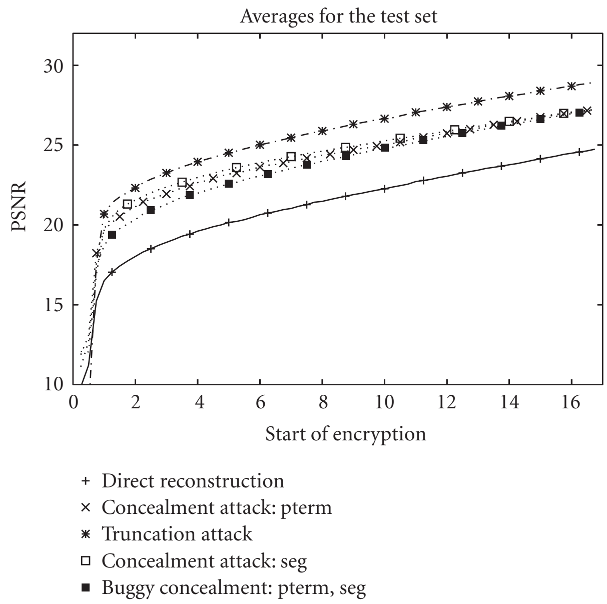


Figure 19: Average results for resolution 256 × 144 and layer progression with a cblk size of 8 × 8.

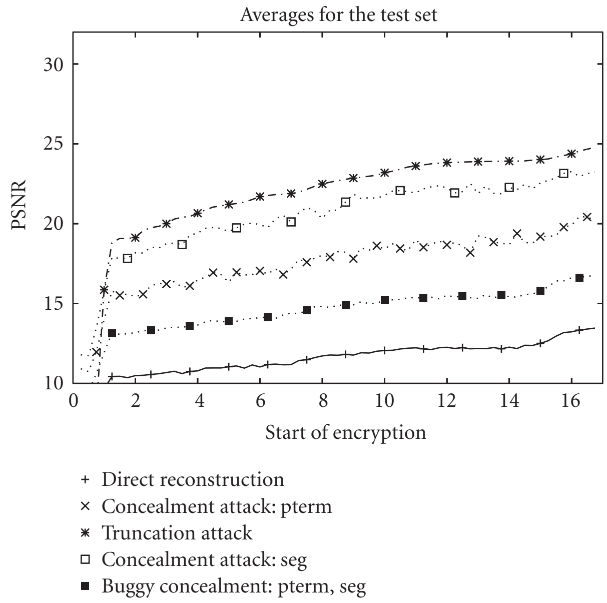


Figure 20: Average results for resolution 256 × 144 and resolution progression with default cblk size.

4.2. *Proposal of a fully format-compliant encryption scheme*

In some cases, fully format-compliant encryption is desired, which allows the complete decoding of the bitstream also for tier2. For example, if SOP and EPH markers are not used (which is the default setting in all reference implementations), then rate adaption is impossible if the packet headers

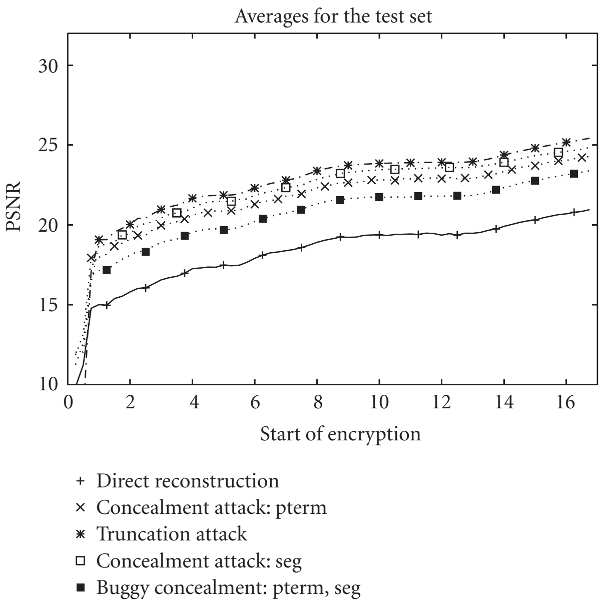
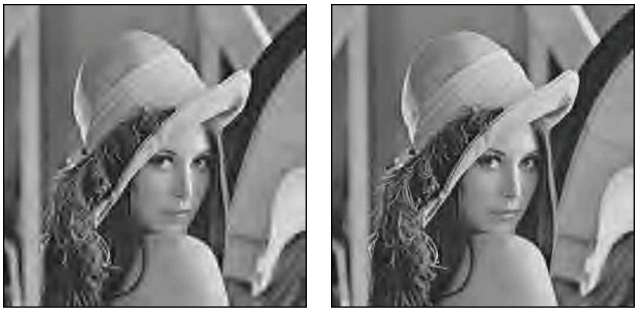


FIGURE 21: Average results for resolution 256 × 144 and resolution progression with a cblk size of 8 × 8.



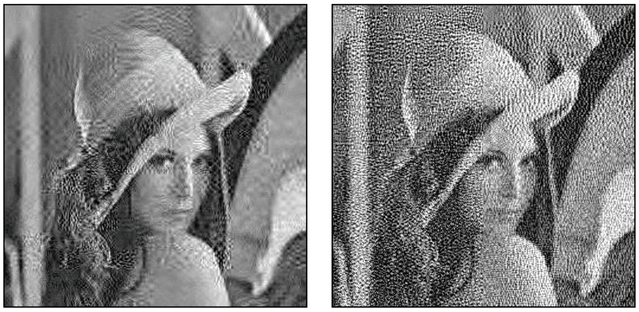
(a) Start position 2%, PSNR 26.9 dB, ESS 0.18

(b) Start position 3%, PSNR 28.4 dB, ESS 0.69

FIGURE 22: Truncation attacks for layer progression.

have been encrypted only in a bitstream-compliant (but not format-compliant) way. In the format-compliant encryption scheme we propose here, the information needed to perform such tasks as rate adaption is kept intact: the distinction between packet body and packet header and the distinction between individual packets are available to the decoder. The information that would be needed to create a strong fingerprint is destroyed.

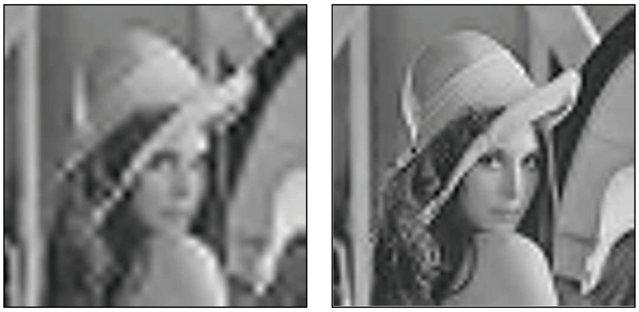
As discussed before, the information contained in the packet header relates to inclusion of codeblocks in the packet, the length of each codeblock, the number of coding passes, and the number of leading zero bitplanes. Each of these pieces of information should be hidden from an attacker. Theoretically, it would be possible to take the packet data and just randomly generate valid packet headers. The resulting bitstream would be format-compliant and decodable. However, reversibility would be hard to realize, as the whole original packet header information would have to be regarded as key material. In a format-compliant encryption approach for packet headers, it should be possible to reconstruct the original packet headers from the encrypted packet header data by the use of a key.



(a) 8 × 8 codeblocks, PSNR 21.9 dB, ESS 0.42

(b) 64 × 64 codeblocks, PSNR 14.37 dB, ESS 0.28

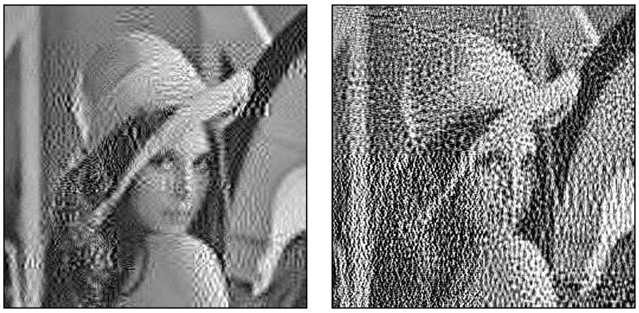
FIGURE 23: Direct reconstruction for layer progression and encryption at 2%.



(a) Start position 2%, PSNR 22.6 dB, ESS 0.43

(b) Start position 3%, PSNR 25.4 dB, ESS 0.58

FIGURE 24: Truncation attacks for resolution progression.



(a) 8 × 8 codeblocks, PSNR 20.26 dB, ESS 0.40

(b) 64 × 64 codeblocks, PSNR 11.28 dB, ESS 0.26

FIGURE 25: Direct reconstructions for resolution progression and encryption at 3%.

In the following, we propose format-compliant transformations for each piece of information contained in the packet header, which prevent an attacker from easily linking an encrypted image to its plaintext version by comparing header information. These transformations make use of a key-dependent random keystream, the knowledge of which allows the decoder to obtain the original packet header. The resulting bitstream is format-compliant.

The proposed key-dependent transformations use permutations. As permutations are principally vulnerable to known-plaintext attacks, the key for the transformation of header information should under no circumstances be derived from the key that is used for the encryption of the packet bodies.

4.2.1. CCP lengths and number of coding passes

JPEG2000 explicitly signals both the number of coding passes and the length of each codeblock contribution. (Depending on the block coding, in other codecs these numbers can be derived from each other, in JPEG2000 they are independent, see [8].) In order to prevent access to this information, we distribute the CCP lengths and number of coding passes in each packet among all nonempty codeblocks. Note that this can happen before or after the transformation of lzb and/or inclusion information. The number of coding passes gives only little information to a potential attacker, the lengths of the codeblock contributions are a more valuable source of information, as they are more distinctive. The permutation algorithm we propose here can be applied to both (for clarity of reading we give the description for lengths only).

We need an approach that is key-dependent and reversible. Adding a random number modulo the total length of all CCPs to the offset of each CCP in the packet is not feasible: some offsets might change their relative positions during this procedure and the decoder would lack information whose random number is associated with which CCP. The situation leads to rather unique requirements for the permutation: given a vector of nonzero positive integers (the CCP-lengths or the number of coding passes) and a random keystream, we want an output vector that randomly redistributes the lengths among all positions (using all possible mappings), preserves the overall sum of the lengths, and has the same number of elements, each of which has to be a nonzero positive integer. Furthermore, the permutation needs to be reversible: given the random keystream and the output vector, the original vector has to be reconstructable.

To achieve a transformation that adheres to these requirements, we change packet lengths (and number of coding passes, resp.) in overlapping pairs, starting with the first and the second length, moving on to the second and the third length, and so on. We redistribute the lengths between a pair of lengths by adding a random number from 0 to the total length of the two packets. This addition is performed modulo the packet size minus one and after the modulo operation, we add one. Thus the size of any packet can never become zero. To avoid that an attacker can obtain the original sum of the pairs, we shuffle the lengths before and after redistributing them. We give the procedure in pseudocode below. `v[]` is a vector of nonzero positive integers (indexing starts at 1). `random()` returns a random float number in $[0, 1)$. `mod` is the modulo operation, which can return a negative residual (as is the case in many programming languages).

```
shuffle (v)
borders := size (v)−1
for i := 1 to borders
    sum := v[i] + v[i + 1]
    r := (int) random () ∗sum
    newBorder := ((v[i] + r) mod (sum− 1)) + 1;
    v[i] := newBorder;
    v[i + 1] := sum − newBorder
end for
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)
    end if
```

and finally unshuffling the result again.

This approach allows to completely redistribute lengths and coding passes among the codeblocks in a packet. The number of possible alterations depends on the number of codeblocks, and how they make their contributions to the individual packets. If a small number of packets contain many contributions, more alterations are possible than if a large number of packets only contain a small number of contributions each. Two nonempty codeblock contributions in a packet are enough to hide their lengths and number of coding passes. Therefore, in practical scenarios, there will always be ample opportunity to sufficiently randomize CCP lengths and number of coding passes.

4.2.2. Leading zero bitplanes

The number of leading zero bitplanes (LZB) for each codeblock is coded by using tag trees [8]. 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).

To transform the number of leading zero bitplanes, we simply use the keystream to generate random bytes. We then add a random byte to the number of leading zero bitplanes modulo a previously determined maximum number of skipped bitplanes. For decoding, the random byte is subtracted instead of being added. The maximum number of skipped bitplanes needs to be signaled to the decoder, for example, by inserting it into the key or by prior arrangement. Note that the maximum number of skipped bitplanes needs to be greater than or equal to the original maximum number of skipped bitplanes (otherwise the modulo operation cannot be reversed). Theoretically, the new number could be arbitrarily high (we found no restrictions in that respect in the standard), but most implementations will have a maximum of bitplanes for the representation of coefficient data that must not be exceeded.

When the number of leading zero bitplanes is changed, the length of the output from the associated tag tree might change as well. This change in length has to be reflected in the tile header, otherwise the decoder will complain. Alternatively, if only a single tile is used, the length in the tile header can be set to be unspecified. Furthermore, the maximum number of bitplanes needed to represent the coefficients in each subband can be derived from information contained in the main header. "The maximum number of bit-planes available for the representation of coefficients in any subband, b, is given by M_b as defined ([9], Equation E.2, page 70)." Equation E.2 in [9] basically derives the number M_b from information contained in the `QCD` and `QCC` marker segments in the main header. Therefore, to achieve full format-compliance, the main header needs to be changed accordingly. Otherwise, decoding will still work, but the decoder might issue a warning. Note, however, that neither one of the reference implementations JJ2000 and Jasper, which we used in our tests, issued a warning.

The total number of possible changes for all packets depends on the number of available codeblocks. If more codeblocks exist, more information can be randomized. The visual information that can potentially be gained by an attacker through the leading zero bitplanes is destroyed by the proposed transformation.

4.2.3. Inclusion information

The inclusion information is an interesting part of the packet header that might leak information to a potential attacker. Each packet contains the inclusion information for a certain quality layer for all codeblocks in the precinct. For the sake of simplicity, we assume that no precinct partitioning is used, so each packet contains inclusion information for all codeblocks of all subbands in the resolution associated with the packet. There are four types of inclusions that codeblock c can have in packet p:

(FI) c is included in p for the first time, that is, c has not been included in any previous packet;
(NI) c is not included in p and has never been included in any previous packet;
(PI) c has been included in a previous packet and is also included in p;
(PN) c has been included in a previous packet but is not included in p.

The sequence of inclusion information of each codeblock is coded depending on the type of inclusion. FI and NI are coded by an inclusion tag-tree. For PI, and PN, that is, previous inclusions, a 1 is coded in the header if the codeblock is included again in the current packet, and a 0 is coded if it is not included in the current packet.

The goal of the proposed approach is 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. We limit the approach to permutation of the original inclusion information and do not split available packet body data from one codeblock to more codeblocks and we also do not merge contributions from distinct codeblocks in a single codeblock. Also, the permutation is applied per packet, that is, we do not merge the packet body data from different packets, as this would have a detrimental effect on the scalability properties of the resulting bitstream. These restrictions do not interfere with the aim to prevent the creation of a strong fingerprint from the sequence of inclusions (although they would help to hide the number of inclusions of each type) and have the advantage of facilitating straightforward reversibility.

We distinguish two kinds of packets: an *empty packet* is a packet for which a header is written, but which does not contain any codeblock contributions, that is, all codeblocks are included as NI or PN. In a *nonempty packet*, there is at least one contribution from one of the codeblocks, that is, at least one codeblock is included as FI or PI.

We define the *inclusion vector* v_p for a packet p as the sequence of the inclusion information for each codeblock associated with p (i.e., each codeblock in the subbands of the resolution associated with p):

$$v_p = [I_{c_i}], \quad \forall c_i \in p,\ I_{c_i} \in \{\mathrm{FI}, \mathrm{NI}, \mathrm{PI}, \mathrm{PN}\}. \tag{1}$$

The order of elements in the inclusion vector follows the order in which the codeblocks are scanned during image coding. In JPEG2000, this is an ordering by subband in the sequence HL, LH, HH followed by a lexicographical ordering over the 2D coordinates of all codeblocks in the subband. Obviously, the order in which subsequent items of the same inclusion type appear are irrelevant, and count as the same permutation. We give the number of possible distinct permutations further below.

An arbitrary full permutation of the inclusion vector of each packet would not produce a format-compliant bitstream. Consider the following example. After the permutation of the inclusion vectors for two packets p_l and p_{l+1} let codeblock c be included in packet p_l as FI. An arbitrary permutation could assign inclusion type NI to c in p_{l+1}. This would lead to a contradiction because c can never be NI after its first inclusion.

Only the first permutation in a resolution r may be a full permutation (but permutations do not necessarily have to start at the packet of the first layer). After the first permutation, the permutations for the subsequent packets have to regard the inclusion information that has been signaled in the directly preceding packet. The transitions in Table 1 are possible.

A codeblock can only be included as FI or NI in p_l if it has been included as NI in packet p_{l-1}. PI and PN can follow a previous inclusion of FI, PI, or PN. It follows that permutations can only be performed for nonempty packets, because for empty packets the positions of the inclusions of types NI and PN are fully determined by the previous packet. Also note that the number of inclusions of type FI plus the number of inclusion of NI in p_l is always equal to the number of inclusions of type NI in p_{l-1}. The same is true for inclusions of type PI and PN: their number in p_l is equal to the number of inclusions of type FI, PI, and PN in p_{l-1}.

We perform the permutations in compliance with these transition rules to produce a (semantically) consistent sequence of inclusion information over the packets of each resolution. First, in each resolution, the first packet suitable for

Table 1

p_l	FI	NI	PI	PN
p_{l-1}	NI	NI	FI, PI, PN	FI, PI, PN

permutation is searched. This packet has to have at least one nonempty codeblock contribution (FI or PI) and one empty inclusion (NI or PN). Packets for which the codeblocks all have the same inclusion information are obviously not suitable for permutation, and packets with mixed FI and PI only occur after the first candidate packet. The inclusion information in the first candidate packet is permuted.

For the subsequent nonempty packets, the inclusion information of the immediately preceding packets is regarded in the permutation. The inclusion vector v for a packet p_l is split into two vectors: $v^{(1)}$ contains all FI and NI inclusions, and $v^{(2)}$ contains all inclusions of types PI and PN. Both vectors are permuted randomly (using the key-stream) to form $\hat{v}^{(1)}$ and $\hat{v}^{(2)}$. According to the possible transitions given above, the elements of $\hat{v}^{(1)}$ are assigned (in the randomized order) to the positions that are marked as NI in the packet of the previous layer, p_{l-1}. The elements of $\hat{v}^{(2)}$ are assigned to the remaining positions (again in the randomized order). The inclusions in p_l that mark a nonempty codeblock contribution, that is, FI and PI, are assigned the length and number of new coding passes of the nonempty codeblock contributions of the correct inclusion vector v in the order in which these contributions appear in v (these CCP lengths and the number of coding passes may be subject to transformation later, or maybe have already been transformed). All first inclusions (FI) in $\hat{v}^{(1)}$ are assigned the number of leading zero-bitplanes in the order of FI-inclusions in v. After all packets have been processed, the new header information is written. If the key that has been used for the permutation is known, this procedure is reversible. Note that the permutation in this approach crosses subband boundaries: the inclusion information is reassigned over all codeblocks in the packet.

For empty packets, no permutation is possible. For these packets, the inclusion information only needs to be updated based on the inclusion information in the previous packet, according to the possible transitions.

To illustrate the process of format-compliant permutation, we give an example. Let $v_{p_l} = [\mathrm{FI}, \mathrm{NI}, \mathrm{NI}]$ be the inclusion vector of the first candidate package p_l in a resolution with three codeblocks c_0, c_1, c_2. After permutation, the new inclusion information is $\hat{v}_{p_l} = [\mathrm{NI}, \mathrm{NI}, \mathrm{FI}]$. The length of the codeblock contribution and the number of leading zero-bitplanes are transferred from c_0 to c_2. In the next packet p_{l+1}, let the real inclusion information be given by $v_{p_{l+1}} = [\mathrm{PI}, \mathrm{FI}, \mathrm{NI}]$. This vector is split into $v^{(1)}_{p_{l+1}} = [\mathrm{FI}, \mathrm{NI}]$ and $v^{(2)}_{p_{l+1}} = [\mathrm{PI}]$. Considering the "faked" inclusion information $\hat{v}_{p_l}$ of the previous packet, the positions of codeblocks c_0 and c_1 are the candidate positions for inclusions of type FI and NI. $v^{(1)}_{p_{l+1}}$ is permuted to form $\hat{v}^{(1)}_{p_{l+1}} = [\mathrm{NI}, \mathrm{FI}]$ and the new inclusion information is assigned to the respective positions of c_0 and c_1. The length of the contribution and the number of coding passes and leading zero-bitplanes are updated for the nonempty contribution. In this example, $v^{(2)}_{p_{l+1}}$ only has one element which has to be assigned to the position of codeblock c_2 to form a consistent sequence of inclusion information. The length of the contribution and the number of coding passes are updated for this codeblock. The new inclusion information for p_{l+1} is $[\mathrm{NI}, \mathrm{FI}, \mathrm{PI}]$.

We now turn to the investigation of the number of possible permutations. For the lower resolutions, there will only typically be few codeblocks that contribute to each packet, so the number of permutations will be very limited. As the permutation of packet headers is only used in conjunction with the encryption of packet bodies, this is not a problem. The fingerprint that could be derived from the lower resolutions is not very distinctive, the main point is to destroy fingerprints in the higher resolutions.

We can give the number of possible permutations for a packet p_l which contains the codeblock contributions for layer l (for a specific resolution). Let $|N_{p_l}|$ be the total number of codeblocks that are relevant for p_l, and $|\mathrm{FI}_{p_l}|, |\mathrm{NI}_{p_l}|, |\mathrm{PI}_{p_l}|, |\mathrm{PN}_{p_l}|$ the number of codeblocks in p_l with inclusion type FI, NI, PI, and PN, respectively. If the p is the first packet to be permuted, we have no restrictions in the possible permutations. Furthermore, in this case, the inclusion information in p will consist of either only FI and NI or of PN and PI (because otherwise a previous packet would have been the first to be permuted). Without loss of generality, we assign the positions for inclusions of types FI and PI. The number of possible permutations is then given as

$$\binom{|C_{p_l}|}{|\mathrm{FI}_{p_l}|}\binom{|C_{p_l}|}{|\mathrm{PI}_{p_l}|}. \tag{2}$$

If p is a packet that pertains to a higher layer than the the first candidate, the inclusion information of the preceding packet p_{l-1} has to be taken into account. Inclusions of types FI and NI in p can go into positions that are included as NI in p_{l-1}. The rest of the positions can be assigned to inclusions of type PI and PN in p_l. The number of possible permutations is determined by

$$\binom{|\mathrm{NI}_{p_{l-1}}|}{|\mathrm{FI}_{p_l}|}\binom{|\mathrm{FI}_{p_{l-1}}| + |\mathrm{PI}_{p_{l-1}}| + |\mathrm{PN}_{p_{l-1}}|}{|\mathrm{PI}_{p_l}|}. \tag{3}$$

The actual number of permutations for a given input image depends on a variety of factors. The used compression parameters influence the number of codeblocks that are available for permutation in the first place. With small codeblock sizes the number of available codeblocks increases. If the number of quality layers is increased, then there are also more packets and therefore more permutations can be performed. The rate with which the image is encoded also influences the number of packets that are included in the codestream. Finally, the number of permutations that can be applied to the packets of a resolution is influenced by the point at which the first candidate packet is found, and how diverse the inclusion information is in this packet and the following packets. If all the codeblocks of a resolution always have the

Table 2: Number of possible format-compliant permutations of the inclusion information for Lena (*wlev* = 5).

CBlk size	Rate (bpp)	Number of layers	Number of permutations
64 × 64	3	32	10^{217}
32 × 32	3	32	10^{938}
64 × 64	3	12	10^{56}
32 × 32	3	12	10^{257}
64 × 64	0.25	32	10^{42}
32 × 32	0.25	32	10^{146}

same inclusion information in each packet, then no permutation of inclusion information is possible. Luckily, this case is extremely unlikely in practice. Table 2 shows the number of possible permutations for the Lena image (512 × 512 pixels) with different compression settings.

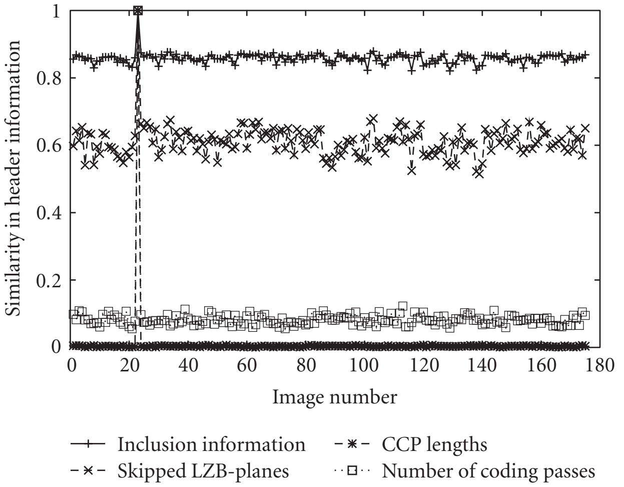


Figure 26: Comparison of similarity in header information for 175 images.

4.2.4. Combined format-compliant header transformation

The format-compliant transformation of the different pieces of information in the packet headers can and should be combined. The format compliance of the combined format-compliant header encryption has been verified experimentally by decoding the encrypted files with the reference implementations Jasper and JJ2000. The order in which they are applied is arbitrary, only decoding has to apply the reverse transformations in the reverse order. The effectiveness of the transformations depend on the settings of the compression parameters, as discussed above. It is noteworthy that the requirement for packet body encryption, namely, to use large codeblocks and few quality layers, can be lifted with packet header encryption. Most of the transformations work better for small codeblock sizes and many quality layers.

With the combination of the transformations, an attacker is prevented from creating a distinctive fingerprint from the header information. To show that the transformations are sufficient to destroy a fingerprint which is based on the details of the packet header, we have tested 175 images, all taken with the same camera model and cropped to 512 × 512 pixels at 8 bpp. All of the images are encoded with JPEG2000 at a bitrate of 0.25 bpp with 32 quality layers and a codeblock size of 32 × 32. The header information in all the packets of each image is recorded. We then compare the header information of a single image from the set to the header information of each other image in the set. The ratio of the number of items in the header information that have the same value (at the same position) to the total number of items is recorded. In the plots, the ordinate shows this value for each class of header information and each image. Note that for CCP lengths and number of coding passes, we ignore positions in the header for which the information is 0 for both the reference and comparison image. Figure 26 shows the similarity in header information of one image (# 23) with the other images. It can be seen that the similarity measures to other images are confined within a certain range of variance. It is not surprising to see that the similarity of the CCP-lengths is very small for differing images. Interestingly, the number of corresponding items in the inclusion information is very large. This is due to the fact that for the inclusion information, we also counted all the inclusions of type NI which at this bitrate occur a lot. The variance of the similarity of the inclusion information is confined relatively strictly and therefore also the inclusion information may serve as a discriminating feature.

Obviously, the situation will be different if the reference image is re-encoded with different compression parameter settings. But as this illustration shows, any class of header information can be used to link a known JPEG2000 plaintext to a packet-body encrypted ciphertext. For many applications that require full confidentiality, such a leak of information constitutes a major compromise of security.

The proposed transformations can be used to prevent the creation of a fingerprint which uses the details of the packet headers, such as those proposed by [27–29]. Figure 27 shows the same comparison as in Figure 26, but this time with the header information of the reference image transformed. It can be observed that the proposed transformation methods obstruct the identification of the image in the set of 175 images: the transformed headers bear no similarity to the original header. Only for the codeblock lengths, a minute trace remains. This is due to packets of the lowest two resolutions which only contain a single codeblock each. For the used compression settings, no transformation was possible for these packets.

Of course, some information remains that can be used to create a (very) weak fingerprint. For example, we do not cross packet boundaries and merge or split the data of packets, so an attacker knows the number of packets and the amount of data each of the packets contains. Furthermore, the number of inclusions of each inclusion type stays the same as in the plaintext image. This information could be used to obtain a fingerprint, albeit a much weaker one than if the order of inclusions was known. If packet boundaries were crossed and furthermore inclusion information was split up among

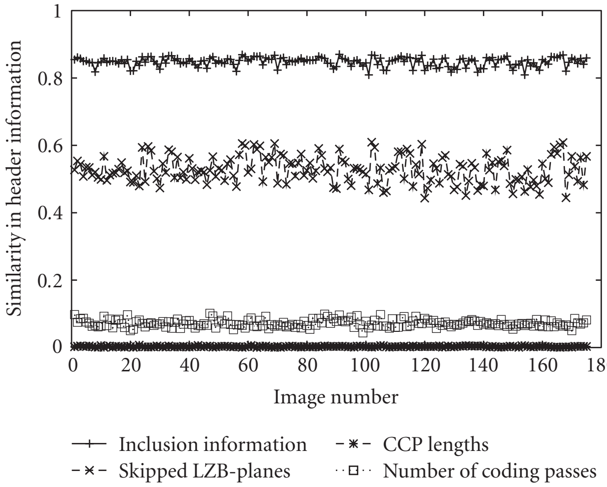


FIGURE 27: Comparison of similarity in header information for transformed header information.

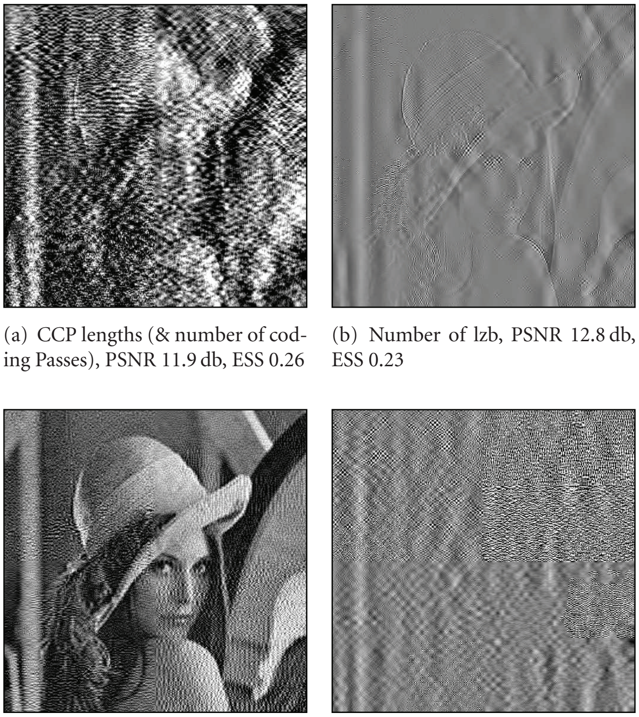
codeblocks, a possible fingerprint would be further weakened. However, the downside would be a loss in semantics for the encrypted version (which would make rescaling more unreliable, e.g.). If format compliance is desired in a sense that allows to perform tasks like rate adaption in the encrypted domain, the information needed for these tasks will always need to be preserved to some extent.

4.2.5. *Visual examples*

In order to give an illustration of the extent of information contained in the header, we give some visual examples of reconstructed plaintext images for which only the packet headers have been encrypted and the packet bodies have been left-unencrypted. As the number of coding passes has only very little impact on the visual quality, we have combined their transformation with the transformation of the CCP lengths. Figure 28 shows the reconstruction for large codeblocks and few quality layers, Figure 29 for medium-sized codeblocks and more quality layers, and Figure 30 for very small codeblocks. It can be observed that the impact of the transformations is increased with the number of codeblocks, especially for inclusion information and leading zero bitplanes (at least visually, PSNR and ESS are not well suited for images of such low quality).

5. CONCLUSION

For packet body-based encryption approaches, compression parameters, especially the codeblock size and the number of quality layers, significantly influence JPEG2000 encryption. A large codeblock size increases the security for application scenarios that strive for confidentiality, while small codeblock sizes are preferred for transparent encryption. Many quality layers increase the information leakage on one side, but are absolutely inevitable for the reduction of complexity of the encryption process. For application scenarios which require a higher level of security, the additional



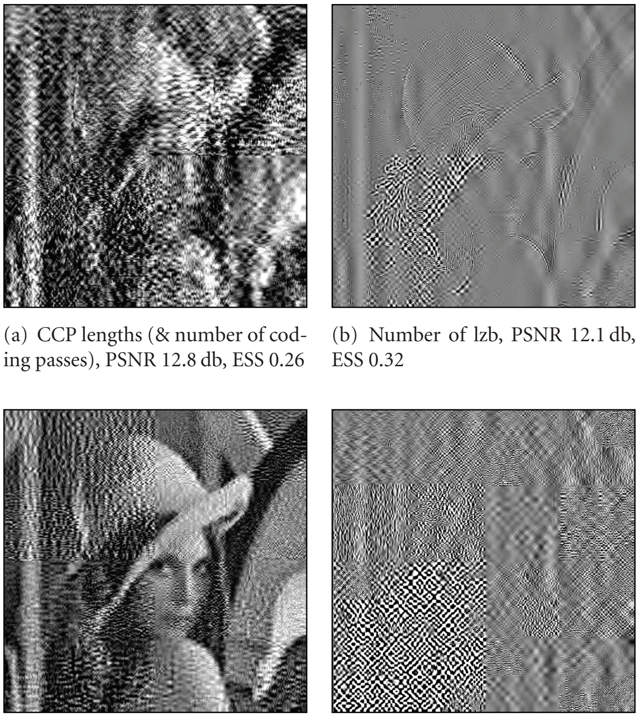
(a) CCP lengths (& number of coding Passes), PSNR 11.9 db, ESS 0.26

(b) Number of lzb, PSNR 12.8 db, ESS 0.23

(c) Inclusion information, PSNR 16.95 db, ESS 0.36

(d) All transformations, PSNR 9.2 db, ESS 0.24

FIGURE 28: Visual examples of reconstructions with transformed packet headers for Lena @ 1 bpp, cblk. size 64 × 64, 16 layers.



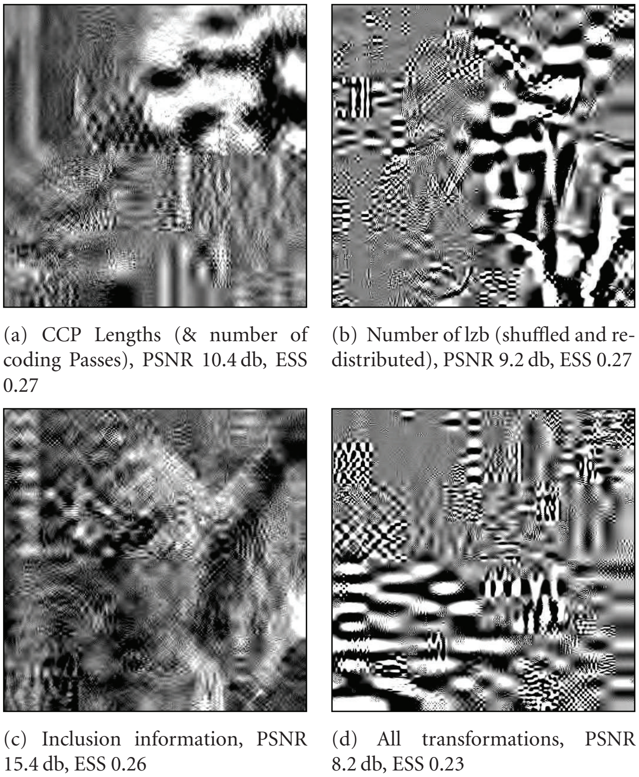
(a) CCP lengths (& number of coding passes), PSNR 12.8 db, ESS 0.26

(b) Number of lzb, PSNR 12.1 db, ESS 0.32

(c) Inclusion information, PSNR 16.0 db, ESS 0.31

(d) All transformations, PSNR 8.5 db, ESS 0.27

FIGURE 29: Visual examples of reconstructions with transformed packet headers for Lena @ 1 bpp, cblk. size 32 × 32, 32 layers.


(a) CCP Lengths (& number of coding Passes), PSNR 10.4 db, ESS 0.27

(b) Number of lzb (shuffled and redistributed), PSNR 9.2 db, ESS 0.27

(c) Inclusion information, PSNR 15.4 db, ESS 0.26

(d) All transformations, PSNR 8.2 db, ESS 0.23

Figure 30: Visual examples of reconstructions with transformed packet headers for Lena @ 1 bpp, cblk. size 8 × 8, 32 layers.

bitstream-compliant encryption of the packet headers is proposed. This scheme increases the security while preserving the simple transcodability. Format-compliant encryption of the packet headers can be used for increasing security by eliminating information leakage of plaintext packet headers. If headers are encrypted in a format-compliant way, the packet borders are known to the decoder. This allows tasks such as rate adaption to be performed directly in the encrypted domain. It has been shown that the codeblock size, the number of quality layers, and the bitrate are the compression parameters that have the most influence on the number of possible transformations for header encryption.

ACKNOWLEDGMENTS

This work has been partially funded by the Austrian Science Fund (FWF) under Projects nos. 15170 and P19159-N13 and by the European Commission through the IST Programme under contract IST-2002-507932 ECRYPT.

REFERENCES

[1] B. Furht and D. Kirovski, Eds., *Multimedia Security Handbook*, CRC Press, Boca Raton, Fla, USA, 2005.

[2] A. Uhl and A. Pommer, *Image and Video Encryption. From Digital Rights Management to Secured Personal Communication*, vol. 15 of *Advances in Information Security*, Springer, New York, NY, USA, 2005.

[3] D. Engel and A. Uhl, "Parameterized biorthogonal wavelet lifting for lightweight JPEG2000 transparent encryption," in *Proceedings of the ACM Multimedia and Security Workshop, (MM-SEC '05)*, pp. 63–70, New York, NY, USA, August 2005.

[4] D. Engel and A. Uhl, "Secret wavelet packet decompositions for JPEG2000 lightweight encryption," in *Proceedings of the 31st International Conference on Acoustics, Speech, and Signal Processing (ICASSP '06)*, vol. 5, pp. 465–468, Toulouse, France, May 2006.

[5] R. Norcen and A. Uhl, "Performance analysis of block-based permutations in securing JPEG2000 and SPIHT compression," in *Prooceedings of the Visual Communications and Image Processing 2005 (VCIP '05)*, S. Li, F. Pereira, H.-Y. Shum, and A. G. Tescher, Eds., vol. 5960 of *Proceedings of SPIE*, pp. 944–952, Beijing, China, July 2005.

[6] ISO/IEC 15444-8, "Final Committee Draft. Information Technology—JPEG2000 image coding system, part 8: secure JPEG2000," Tech. Rep., ISO, 2004.

[7] ISO/IEC 15444-8, "Information technology—JPEG2000 image coding system, part 8: secure JPEG2000," April 2007.

[8] D. Taubman and M. W. Marcellin, *JPEG2000—Image Compression Fundamentals, Standards and Practice*, Kluwer Academic, Dordrecht, The Netherlands, 2002.

[9] ISO/IEC 15444-1, "Information technology—JPEG2000 image coding system, part 1: core coding system," December 2000.

[10] ISO/IEC 15444-4, "Information technology—JPEG2000 image coding system, part 4: conformance testing," December 2004.

[11] Y. Mao and M. Wu, "Security evaluation for communication-friendly encryption of multimedia," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '04)*, vol. 1, pp. 569–572, Singapore, October 2004.

[12] F. Dufaux, S. Wee, J. Apostolopoulos, and T. Ebrahimi, "JPSEC for secure imaging in JPEG2000," in *Applications of Digital Image Processing XXVII*, I. A. G. Tescher, Ed., vol. 5558 of *Proceedings of SPIE*, pp. 319–330, August 2004.

[13] J. Apostolopoulos, F. Dufaux, S. Wee, T. Ebrahimi, Q. Sun, and Z. Zhang, "The emerging JPEG2000 security (JPSEC) standard," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '06)*, May 2006.

[14] ISO/IEC 15444-1, "Final Committee Draft, Information Technology—JPEG2000 image coding system, part 1: core coding system," Tech. Rep., ISO, March 2000.

[15] R. Grosbois, P. Gerbelot, and T. Ebrahimi, "Authentication and access control in the JPEG2000 compressed domain," in *Proocedings of the Applications of Digital Image Processing XXIV*, A. G. Tescher, Ed., vol. 4472 of *Proceedings of SPIE*, pp. 95–104, San Diego, Calif, USA, July 2001.

[16] R. Norcen and A. Uhl, "Selective encryption of the JPEG2000 bitstream," in *Proceedings of the Communications and Multimedia Security of Lecture Notes on Computer Science, (CMS '03)*, A. Lioy and D. Mazzocchi, Eds., vol. 2828 of *Proceedings of the IFIP TC6/TC11 6th Joint Working Conference on Communications and Multimedia Security*, pp. 194–204, Turin, Italy, October 2003.

[17] F. Dufaux and T. Ebrahimi, "Securing JPEG2000 compressed images," in *Proocedings of the Applications of Digital Image Processing XXVI*, I. A. G. Tescher, Ed., vol. 5203 of *Prooceedings of SPIE*, pp. 397–406, November 2003.

[18] H. Kiya, D. Imaizumi, and O. Watanabe, "Partial-scrambling of images encoded using JPEG2000 without generating

marker codes," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '03)*, vol. 3, pp. 205–208, Barcelona, Spain, September 2003.

[19] M. Wu and V. Mao, "Communication-friendly encryption of multimedia," in *Proceedings of the IEEE Multimedia Signal Processing Workshop (MMSP '02)*, St. Thomas, Virgin Islands, USA, December 2002.

[20] V. Conan, Y. Sadourny, and S. Thomann, "Symmetric block cipher based protection: Contribution to JPSEC," ISO/IEC JTC 1/SC 29/WG 1 N 2771, October 2003.

[21] H. Wu and D. Ma., "Efficient and secure encryption schemes for JPEG2000," in *Proceedings of the 2004 International Conference on Acoustics, Speech and Signal Processing (ICASSP '04)*, vol. 5, pp. 869–872, May 2004.

[22] Y. Wu and R. H. Deng, "Compliant encryption of JPEG2000 codestreams," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '04)*, vol. 2, pp. 3439–3442, Singapore, October 2004.

[23] J. Fang and J. Sun, "Compliant encryption scheme for JPEG2000 image code streams," *Journal of Electronic Imaging*, vol. 15, no. 4, Article ID 043013, 4 pages, 2006.

[24] M. Grangetto, E. Magli, and G. Olmo, "Multimedia selective encryption by means of randomized arithmetic coding," *IEEE Transactions on Multimedia*, vol. 8, no. 5, pp. 905–917, 2006.

[25] T. Stütz and A. Uhl, "On format-compliant iterative encryption of JPEG2000," in *Proceedings of the 8th IEEE International Symposium on Multimedia (ISM '06)*, pp. 985–990, Los Alamitos, Calif, USA, 2006.

[26] Digital Cinema Initiatives & LLC (DCI), "Digital cinema system specification v1.1.," http://www.dcimovies.com/DCI_DCinema_System_Spec_v1_1.pdf, April 2007.

[27] C. Liu and M. Mandal, "Fast image indexing based on JPEG2000 packet header," in *Proceedings of the 2001 ACM Workshops on Multimedia: Multimedia Information Retrieval*, ACM Press, pp. 46–49, New York, NY, USA, October 2001.

[28] A. Descampe, P. Vandergheynst, C. De Vleeschouwer, and B. Macq, "Coarse-to-fine textures retrieval in the JPEG 2000 compressed domain for fast browsing of large image databases," in *Proceedings of the Multimedia Content Representation, Classification and Security (MRCS '06)*, I. B. Günsel, A. K. Jain, A. M. Tekalp, and B. Sankur, Eds., vol. 4105 of *Lecture Notes in Computer Science*, pp. 282–289, Springer, New York, NY, USA, September 2006.

[29] A. Tabesh, A. Bilgin, K. Krishnan, and M. W. Marcellin, "JPEG2000 and motion JPEG2000 content analysis using codestream length information," in *Proceedings of the Data Compression Conference (DCC '05)*, pp. 329–337, March 2005.

[30] B. M. Macq and J. Quisquater, "Cryptology for digital TV broadcasting," *Proceedings of the IEEE*, vol. 83, no. 6, pp. 944–957, 1995.

[31] A. Uhl and Ch. Obermair, "Transparent encryption of JPEG2000 bitstreams," in *Proceedings of the EC-SIP-M 2005 (5th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services)*, P. Podhradsky, et al., Eds., pp. 322–327, Smolenice, Slovak Republic, 2005.

EURASIP Journal on Wireless Communications and Networking

Special Issue on OFDMA Architectures, Protocols, and Applications

Call for Papers

Orthogonal frequency-division multiple access (OFDMA) technologies are currently attracting intensive attention in wireless communications to meet the ever-increasing demands arising from the explosive growth of Internet, multimedia, and broadband services. OFDMA-based systems are able to deliver high data rate, operate in the hostile multipath radio environment, and allow efficient sharing of limited resources such as spectrum and transmit power between multiple users. OFDMA has been used in the mobility mode of IEEE 802.16 WiMAX, is currently a working specification in 3GPP Long Term Evolution downlink, and is the candidate access method for the IEEE 802.22 "wireless regional area networks." Clearly, recent advances in wireless communication technology have led to significant innovations that enable OFDMA-based wireless access networks to provide better quality-of-service (QoS) than ever with convenient and inexpensive deployment and mobility.

However, regardless of the technology used, OFDMA networks must not only be able to provide reliable and high quality broadband services, but also be implemented cost-effectively and be operated efficiently. OFDMA presents many of the advantages and challenges of OFDM systems for single users, and the extension to multiple users introduces many further challenges and opportunities, both on the physical layer and at higher layers. These requirements present many challenges in the design of network architectures and protocols, which have motivated a significant amount of research in the area. Also, many critical problems associated with the applications of OFDMA technologies in future wireless systems are still looking for efficient solutions. The aim of this special issue is to present a collection of high-quality research papers that report the latest research advances in this field from physical and network layers to practical applications. Original papers are solicited in all aspects of OFDMA techniques including physical layer issues, architectures, protocol designs, enabling technologies, theoretical studies, practical applications, and experimental prototypes. Topics of interest include, but are not limited to:

- Adaptive coding and modulation
- Signal processing for OFDMA
- Interference control techniques
- Bandwidth and resources allocation
- Efficient MAC protocol development
- Routing algorithms and congestion control schemes
- MAC and network layer management
- Cross-layer design and optimization
- Cooperative and game theoretic analysis
- Quality of service provisioning
- Network modeling and performance analysis
- Security and privacy management
- Broadband Wireless Access
- Testbed, experiment, implementation, standards, and practical applications

Authors should follow the EURASIP Journal on Wireless Communications and Networking manuscript format described at the journal site http://www.hindawi.com/journals/wcn/. Prospective authors should submit an electronic copy of their complete manuscript through the journal Manuscript Tracking System at http://mts.hindawi.com/, according to the following timetable.

Manuscript Due	August 1, 2008
First Round of Reviews	November 1, 2008
Publication Date	February 1, 2009

Guest Editors

Victor C.M. Leung, The University of British Columbia, 2329 West Mall Vancouver, BC, Canada, V6T 1Z4; vleung@ece.ubc.ca

Alister G. Burr, Department of Electronics, University of York, York, YO10-5DD, UK; alister@ohm.york.ac.uk

Lingyang Song, Wireless Group, Philips Research Laboratories, Cross Oak Lane, Redhill, Surrey RH1 5HA, UK; lingyang.song@philips.com

Yan Zhang, Simula Research Laboratory, 1325 Lysaker, Norway; yanzhang@ieee.org

Thomas Michael Bohnert, Siemens AG, 80312 Munich, Germany; thomas.bohnert@siemens.com

Hindawi Publishing Corporation
http://www.hindawi.com

EURASIP Journal on Information Security

Special Issue on Secure Steganography in Multimedia Content

Call for Papers

Steganography, the art and science of invisible communication, aims to transmit information that is embedded invisibly into carrier data. Different from cryptography it hides the very existence of the secret. Its main requirement is undetectability, that is, no method should be able to detect a hidden message in carrier data. This also differentiates steganography from watermarking where the secrecy of hidden data is not required. Watermarking serves in some way the carrier, while in steganography, the carrier serves as a decoy for the hidden message.

The theoretical foundations of steganography and detection theory have been advanced rapidly, resulting in improved steganographic algorithms as well as more accurate models of their capacity and weaknesses.

However, the field of steganography still faces many challenges. Recent research in steganography and steganalysis has far-reaching connections to machine learning, coding theory, and signal processing. There are powerful blind (or universal) detection methods, which are not fine-tuned to a particular embedding method, but detect steganographic changes using a classifier that is trained with features from known media. Coding theory facilitates increased embedding efficiency and adaptiveness to carrier data, both of which will increase the security of steganographic algorithms. Finally, both practical steganographic algorithms and steganalytic methods require signal processing of common media like images, audio, and video. The field of steganography still faces many challenges, for example,

- how could one make benchmarking steganography more independent from machine learning used in steganalysis?
- where could one embed the secret to make steganography more secure? (content adaptivity problem).
- what is the most secure steganography for a given carrier?

Material for experimental evaluation will be made available at http://dud.inf.tu-dresden.de/~westfeld/rsp/rsp.html.

The main goal of this special issue is to provide a state-of-the-art view on current research in the field of steganographic applications. Some of the related research topics for the submission include, but are not limited to:

- Performance, complexity, and security analysis of steganographic methods
- Practical secure steganographic methods for images, audio, video, and more exotic media and bounds on detection reliability
- Adaptive, content-aware embedding in various transform domains
- Large-scale experimental setups and carrier modeling
- Energy-efficient realization of embedding pertaining encoding and encryption
- Steganography in active warden scenario, robust steganography
- Interplay between capacity, embedding efficiency, coding, and detectability
- Steganalytic application in steganography benchmarking and digital forensics
- Attacks against steganalytic applications
- Information-theoretic aspects of steganographic security

Authors should follow the EURASIP Journal on Information Security manuscript format described at the journal site http://www.hindawi.com/journals/is/. Prospective authors should submit an electronic copy of their complete manuscript through the journal Manuscript Tracking System at http://mts.hindawi.com/ according to the following timetable:

Manuscript Due	August 1, 2008
First Round of Reviews	November 1, 2008
Publication Date	February 1, 2009

Guest Editors

Miroslav Goljan, Department of Electrical and Computer Engineering, Watson School of Engineering and Applied Science, Binghamton University, P.O. Box 6000, Binghamton, NY 13902-6000, USA; mgoljan@binghamton.edu

Andreas Westfeld, Institute for System Architecture, Faculty of Computer Science, Dresden University of Technology, Helmholtzstraße 10, 01069 Dresden, Germany; westfeld@inf.tu-dresden.de

Hindawi Publishing Corporation
http://www.hindawi.com

EURASIP Journal on Advances in Signal Processing

Special Issue on Game Theory in Signal Processing and Communications

Call for Papers

Game theory is a branch of mathematics aimed at the modeling and understanding of rational behavior in strategic situations. In the last decade, game theory has been applied to solve conflict problems in economics, and has found important applications in politics, sociology, psychology, and transportation. Game theory has more recently been employed to model and analyze modern communication systems, such as power control in wireless networks and routing in wire line networks. Also, it provides a structured approach to many important signal processing problems, including cognitive radio, waveform design, and dynamic spectrum access. Game theory is successfully applied to design decentralized algorithms and robust signal processing methods in various deployment scenarios.

This special issue aims to promote the field of game theory to the signal processing audience. We are soliciting high-quality unpublished research papers addressing the theory and practice of game theory in signal processing and communications. Topics include, but are not limited to:

- Static non-cooperative games (Nash and Stackelberg equilibria)
- Finite and infinite dynamic games
- Cooperative (bargaining) game theory
- Auctions, coalitions, and pricing
- Game theory for resource allocation in communications
- Game theory for adaptive waveform design
- Game theory for cognitive radio and dynamic spectrum access
- Stochastic games, repeated games, and fading channels
- Development of decentralized algorithms using game theory

Authors should follow the EURASIP Journal on Advances in Signal Processing manuscript format described at the journal site http://www.hindawi.com/journals/asp/. Prospective authors should submit an electronic copy of their complete manuscript through the EURASIP JASP Manuscript Tracking System at http://mts.hindawi.com/, according to the following timetable:

Manuscript Due	October 1, 2008
First Round of Reviews	January 1, 2009
Publication Date	April 1, 2009

Guest Editors

Holger Boche, Berlin Institute of Technology, 10623 Berlin, Germany; boche@hhi.fhg.de

Zhu Han, Department of Electrical and Computer Engineering, College of Engineering, Boise State University Boise, ID 83725, USA; zhuhan@boisestate.edu

Erik G. Larsson, Division of Communication Systems, Department of Electrical Engineering (ISY), Linköping University, 581 83 Linköping, Sweden; erik.larsson@isy.liu.se

Eduard A. Jorswieck, Communications Laboratory, Dresden University of Technology, 01062 Dresden, Germany; jorswieck@ifn.et.tu-dresden.de

Hindawi Publishing Corporation
http://www.hindawi.com

International Journal of Digital Multimedia Broadcasting

Special Issue on
Broadcasting and Telecommunications at Crossroads: Impacts of NGN and Web 2.0 Concepts on the Future of IPTV

Call for Papers

Broadcasting, telecommunications, and the Internet are converging. As transporting the traditional broadcast TV just over the Internet is probably not generating significant benefits for the customers, IPTV contemporary deployments provide additional capabilities, such as video-on-demand, network-based personal video recorders, time-shifted TV, and so forth. However, key values of IPTV probably originate from the availability of back control channel, allowing for interactive TV, as well as the integration with emerging multimedia group communication services, such as buddy lists featuring presence information and instant messaging, voice over IP, emailing, video conferencing and many more, enabling a new community TV experience. In this context, the notion of *IP multimedia subsystem- (IMS-) based IPTV* is currently gaining momentum in the international NGN and IPTV standardization, by providing the middleware to glue network intelligence and directly to the home or on the move, that is, wire or wireless, respectively, TV service provision.

In light of *Web 2.0,* The internet has become a user-generated multimedia content distribution medium for many different communities, including different types of video services: for example, YouTube, Apple Podcasts, or P2P video architectures like Joost. As the Internet is currently challenging many business cases for the provision of NGN communication services, it will also do so in the field of broadcasting and TV. There is no doubt that in the future NGN- and Web 2.0-enabled IPTV environments will be in a position to provide the required flexibility for supporting adequately the converged markets.

Authors should follow the International Journal of Digital Multimedia Broadcasting manuscript format described at the journal site http://www.hindawi.com/journals/ijdmb/. Prospective authors should submit an electronic copy of their complete manuscript through the journal Manuscript Tracking System at http://mts.hindawi.com/ according to the following timetable:

Manuscript Due	April 1, 2008
First Round of Reviews	July 1, 2008
Publication Date	October 1, 2008

Guest Editors

Thomas Magedanz, Fraunhofer Institute for Open Communication Systems (FOKUS), Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany; thomas.magedanz@fokus.fraunhofer.de

Heinrich Arnold, Innovation Development Laboratory, Deutsche Telekom, Ernst-Reuter-Platz 7, 10587 Berlin, Germany; heinrich.arnold@telekom.de

Pete Distler, Sprint Mobile Media Network, 6220 Sprint Parkway, Overland Park, KS 66251, USA; peter.distler@sprint.com

Ravi S. Sharma, Wee Kim Wee School of Communication and Information, Nanyang Technological University, 31 Nanyang Link, Singapore 637718; asrsharma@ntu.edu.sg

Hindawi Publishing Corporation
http://www.hindawi.com

EURASIP Journal on Advances in Signal Processing

Special Issue on CNN Technology for Spatiotemporal Signal Processing

Call for Papers

A cellular neural/nonlinear network (CNN) is any spatial arrangement of mainly locallycoupled cells, where each cell has an input, an output, and a state that evolves according to some prescribed dynamical laws. CNN represents a paradigm for nonlinear spatial-temporal dynamics and the core of the cellular wave computing (also called CNN technology). Partial differential equations (PDEs) or wave-like phenomena are the computing primitives of CNN. Besides, their suitability for physical implementation due to their local connectivity makes CNNs very appropriate for high-speed parallel signal processing.

Early CNN applications were mainly in image processing. The possible availability of cellular processor arrays with a high number of processing elements opened a new window for the development of new applications and the recovery of techniques traditionally conditioned by the slow speed of conventional computers. Let us name as example image processing techniques based on active contours or active wave propagation, or applications within the medical image processing framework (echocardiography, retinal image processing, etc.) where fast processing provides new capabilities for medical disease diagnosis.

On the other hand, emerging applications exploit the complex spatiotemporal phenomena exhibited by multilayer CNN and extend to the modelling of neural circuits for biological vision, motion, and higher brain function.

The aim of this special issue is to bring forth the synergy between CNN and spatiotemporal signal processing through new and significant contributions from active researchers in these fields. Topics of interest include, but are not limited to:

- Theory of cellular nonlinear spatiotemporal phenomena
- Analog-logic spatiotemporal algorithms
- Learning & design
- Bioinspired/neuromorphic arrays
- Physical VLSI implementations: integrated sensor/processor/actuator arrays
- Applications including computing, communications, and multimedia
- Circuits, architectures and systems in the nanoscale regime
- Other areas in cellular neural networks and array computing

Authors should follow the EURASIP Journal on Advances in Signal Processing manuscript format described at http://www.hindawi.com/journals/asp/. Prospective authors should submit an electronic copy of their complete manuscript through the journal Manuscript Tracking System at http://mts.hindawi.com/, according to the following timetable:

Manuscript Due	September 15, 2008
First Round of Reviews	December 15, 2008
Publication Date	March 15, 2009

Guest Editors

David López Vilariño, Departamento de Electrónica y Computación, Facultad de Fisica, Universidad de Santiago de Compostela, 15782 Santiago de Compostela, Spain; dlv@dec.usc.es

Diego Cabello Ferrer, Departamento de Electrónica y Computación, Facultad de Fisica, Universidad de Santiago de Compostela, 15782 Santiago de Compostela, Spain; diego@dec.usc.es

Victor M. Brea, Departamento de Electrónica y Computación, Facultad de Fisica, Universidad de Santiago de Compostela,
15782 Santiago de Compostela, Spain; victor@dec.usc.es

Ronald Tetzlaff, Lehrstuhl für Grundlagen der Elektrotechnik, Fakultät für Elektrotechnik und Informationstechnik, Technische Universität Dresden, Mommsenstraße 12, 01069 Dresden, Germany; r.tetzlaff@iap.uni-frankfurt.de

Chin-Teng Lin, National Chiao-Tung University, Hsinchu 300, Taiwan; ctpeter.lin@msa.hinet.net

Hindawi Publishing Corporation
http://www.hindawi.com

International Journal of Digital Multimedia Broadcasting

Special Issue on Personalization of Mobile Multimedia Broadcasting

Call for Papers

In recent years, the widespread adoption of multimedia computing, the deployment of mobile and broadband networks, and the growing availability of cheap yet powerful mobile have converged to gradually increase the range and complexity of mobile multimedia content delivery services for devices such as PDAs and cell phones. Basic multimedia applications are already available for current generation devices, and more complex broadcasting services are under development or expected to be launched soon, among which mobile and interactive television (ITV). Among the many challenging issues opened by these developments is the problem of personalization of such services: adaptation of the content to the technical environment of the users (device and network type) and to their individual preferences, providing personalized assistance for selecting and locating interesting programes among an overwhelming number of proposed services.

This special issue is intended to foster state-of-the-art research contributions to all research areas either directly applying or contributing to solving the issues related to digital multimedia broadcasting personalization. Topics of interest include (but are not limited to):

- Mobile TV
- Mobile multimedia broadcasting personalization
- Interactive broadcasting services/interactive television
- Personalization and multimedia home platform (MHP)
- Multimedia content adaptation for personalization
- User behavior and usage modelling
- Standards for modelling and processing (MPEG-21, CC/PP, etc.)
- Personalization issues in DVB-H, DMB, MediaFLO, CMMB, MBMS, and other systems
- Mobile web initiative
- Personalized multimedia and location-based services
- Security and digital rights management
- Applications for personalized mobile multimedia broadcasting with cost-effective implementation

Authors should follow the International Journal of Digital Multimedia Broadcasting manuscript format described at the journal site http://www.hindawi.com/journals/ijdmb/. Prospective authors should submit an electronic copy of their complete manuscript through the journal Manuscript Tracking System at http://mts.hindawi.com/ according to the following timetable:

Manuscript Due	March 1, 2008
First Round of Reviews	June 1, 2008
Publication Date	September 1, 2008

Guest Editors

Harald Kosch, University of Passau, 94030 Passau, Germany; harald.kosch@uni-passau.de

Jörg Heuer, Siemens AG, 80333 Munich, Germany; joerg.heuer@siemens.com

Günther Hölbling, University of Passau, 94030 Passau, Germany; guenther.hoelbling@uni-passau.de

László Böszörményi, University Klagenfurt, 9020 Klagenfurt, Austria; laszlo@itec.uni-klu.ac.at

David Coquil, University of Passau, 94030 Passau, Germany; david.coquil@uni-passau.de

Hindawi Publishing Corporation
http://www.hindawi.com