

# JPEG 2000 BACKWARD COMPATIBLE ERROR PROTECTION WITH REED-SOLOMON CODES

*Didier Nicholson<sup>1</sup>, Catherine Lamy-Bergot<sup>1</sup>, Xavier Naturel<sup>1</sup>, Charly Poulliat<sup>2</sup>*

<sup>1</sup>THALES Communications France,  
TRS/TSI, Colombes.

<sup>2</sup>ENSEA-Cnrs UMR8051-Univ. Cergy Pontoise,  
Equipe Traitement des Images et du Signal

## ABSTRACT

In this paper, a backward compatible header error protection mechanism is described. It consists of the addition of a dedicated marker segment to a JPEG 2000 codestream, that will contain the error correction data generated by a block error correction code (e.g. a Reed Solomon code). This mechanism allows to leave the original data intact, hence providing backward compatibility with the already standardised JPEG 2000. Neither side information from higher level, nor extra signalling encapsulation is needed, as the required information is directly embedded in the codestream and also protected.

## 1. INTRODUCTION

During the establishment of JPEG 2000 standard [1], a set of error resilience tools have been selected, for the transmission of JPEG 2000 compressed images in an error prone environment. Two types of tools are available, on the packet level, which enable synchronisation, and on the entropy coding level, enabling error detection. An analysis of the performances of such tools has been done in [2] and [3]. These tools are however based on one major hypothesis, namely that the headers (Main Header and Tile-part(s) header(s)) of the codestream syntax are guaranteed to be error free. However, in the case of error within the headers, the codestream is not decodable in a proper way, which might conduct to a decoder application crash. The worse is that, generally, it might not be possible to guarantee that the headers will be kept free of errors in many applications.

In order to extend the error resilience to headers and avoid (or strongly limit) the decoding crashes due to error presence, it is consequently necessary to define an error protection mechanism for protecting Main Header and Tile-part(s) header(s). It is to be noted that this protection can be extended to the packet headers when they are

relocated within the Main and Tile-part(s) header(s) thanks to the packed packets functionality of JPEG 2000. Such a protection can either be provided by an external coding scheme, such as the unequal error protection solutions proposed in [4]-[5], or directly be supplied by an embedded mechanism in the JPEG 2000 syntax. The drawbacks of external schemes are well known: in particular, the error correcting scheme must whether have full knowledge of the bistream syntax (resulting in the necessity to transmit extra information between the two coders), or protect it partially blindly (*i.e.* less efficiently, for instance by using statistical ratios to determine the various bitstream parts). Moreover, such schemes cannot take advantage of the various source coder functionalities such as the headers reordering.

As a consequence, it is proposed in this paper to develop a scheme that embeds the protection within the JPEG 2000 stream. Such a mechanism can be in practice whether compatible with JPEG 2000 or not. It was chosen to present here a backward compatible mechanism.

The work presented in this paper has been submitted to the Part 11 of JPEG 2000 Ad Hoc Group, named Wireless JPEG 2000 [6]. This JPEG 2000 Ad Hoc Group is considering the wireless application needs for JPEG 2000, and will standardise the mechanisms enabling advanced error correction and/or handling of JPEG 2000 compressed images.

This paper is organised as follows: Section 2 presents the scheme proposed to ensure error protection of the JPEG 2000 codestream, describing the proposed new marker and its functionalities. Section 3 presents the simulation conditions and the corresponding results obtained. Finally some conclusions are drawn in Section 4 and perspectives are presented.

## 2. A JPEG 2000 BACKWARD COMPATIBLE ERROR PROTECTION SCHEME FOR HEADERS

In this section, the structure of a new marker segment for JPEG 2000 is proposed, that will allow to perform header protection. In a first step, elements on JPEG 2000 syntax, on the implications of the wished backward compatibility

and on error correction codes are given. Then, the marker segment syntax is detailed and default error correction codes are proposed.

### 2.1. JPEG 2000 codestream syntax

A JPEG 2000 compressed image uses markers and marker segments to delimit and signal the compressed information, organised in headers (Main and Tile-parts) and packets. This modular organisation allows flexible bitstream organisation for progressive data representation, such as quality progressive and resolution progressive data progression. A JPEG 2000 codestream always starts by the Main Header followed by one or several Tile-part Headers, each of them followed by compressed data packets, and ends by an End Of Codestream (EOC), as shown in Fig. 1.



Fig. 1. JPEG 2000 codestream structure.

For more information on JPEG 2000 standard, please refer to [1].

### 2.2. Backward compatibility requirements

The objective being to obtain a codestream compliant with JPEG 2000 Part 1 specifications [1] after the insertion of the redundant information, it is necessary to place this information in such a way that any JPEG 2000 Part 1 decoder won't try to interpret it. A solution for this is to insert the redundant information in a dedicated marker segment. This marker will be denoted by EPB for Error Protection Block in the rest of this paper. A JPEG 2000 Part 1 compliant decoder will then skip the (from itself) unknown marker segment and be oblivious to the added data, whereas a JPEG 2000 Part 11 compliant decoder will be able to interpret and use the redundancy for header protection.

The conditions for such a mechanism to work in the context of JPEG 2000 are:

- that the decoder is able to locate the redundant information data block in the codestream without generating complex data indexing mechanism (that would also have to be protected against errors) nor with modifying the first marker segments imposed for the backward compatibility;
- that the marker itself and its length are included in the data range to be protected;
- that a defined block error code is used to protect at least up to the Error Protection Block marker

segment parameters data (which, as later are proposed to be the marker, Length, index, EPB parameters), allowing the recovery of the marker boundaries.

To meet the first of these requirements, a solution is to place the marker segment with the redundant information immediately after the mandatory markers, that is to say:

- the SOC and SIZ marker segments for the Main header;
- the SOT marker for the Tile-part header.

Then, the use of a forward error-correction mechanism such as the one proposed in the following section will ensure that these two conditions are verified. As, for backward compatibility reasons, the original data must be kept untouched, and the redundant information must be concatenated to it and located in a dedicated JPEG 2000 marker segment, a systematic error correction mechanism shall be used.

### 2.3. Forward Error correction mechanism

Error correction and detection codes are traditionally used to provide forward error correction capabilities in error prone environments. Considering that JPEG and JPEG 2000 codestreams are byte aligned, it is especially interesting to work with the Galois Field  $GF(2^8)$  to provide error-correction capability. A well-known and well-suited family of systematic codes in this context is the Reed-Solomon (RS) one [7]. In the following, we will consider the example of RS codes as our FEC codes for header protection, and denote them by  $RS(N,K)$ , where  $N$  is the codewords symbol length and  $K$  the number of information symbols.

The  $RS(N,K)$  applied to  $K$  bytes will generate  $N-K$  redundancy bytes, that may be placed after the  $K$  original (systematic) bytes, this process being applied as long as necessary, as illustrated in Fig. 2.

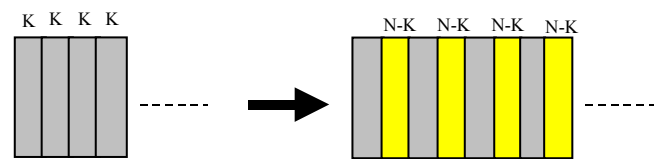


Fig. 2. Example of redundancy generation for an  $RS(N,K)$  code.

In order to remain compliant with JPEG 2000 Part 1 specifications, it was chosen to place the EPB marker segment immediately after the mandatory bytes of the considered header. Fig. 3 illustrates this disposition in the Main header case. Those mandatory bytes, together with the beginning of the EPB (in practice, the marker and its length information) constitutes then a first set of data, whose  $L_1$  bytes is protected by  $L_2$  redundancy bytes

placed immediately after (in the EPB marker data). This way, the bitstream remains compliant to JPEG 2000 Part 1 and if the error-correcting code used for protection is fixed, the decoding can take place without requiring the transfer of an extra (and unprotected) information. This L1+L2 byte section can for instance be generated by an RS(L1+L2,L1) code.

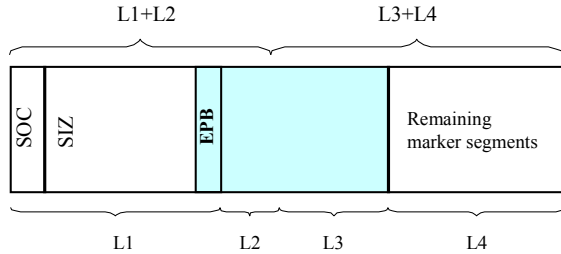


Fig. 3. Position of the EPB marker in the JPEG 2000 codestream in the Main Header case.

The remaining marker segments (L4 bytes) can then be protected by the remaining L3 bytes of the EPB marker segment, for instance with an RS(L3+L4,L4) code.

#### 2.4. Proposed EPB marker segment description

The proposed Error Protection Block (EPB) marker segment contains information about the error protection parameters and data used to protect the headers against errors. The EPB syntax allows to use more than one EPB marker segments in a header, providing then a great flexibility in terms of size and redundancy, and to define future additional error correcting codes.

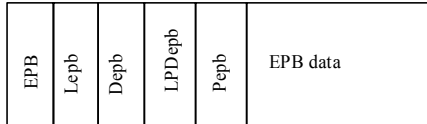


Fig. 4. Description of the proposed Error Protection Block marker segment syntax.

Fig. 4 describes the syntax proposed for the EPB marker segment, where the various fields of the marker are defined as follows:

**EPB:** marker code (Length 16 bits);

**Lepb:** length of marker segment in bytes (not including the marker) (Length 16 bits);

**Depb:** EPB style (for example defines if the current EPB is the latest in the current header) (Length 8 bits);

**LDPepb:** length of the data to be protected by the redundant information (EPB data) carried within the current EPB (Length 32 bits);

**Pepb:** EPB Parameters: This defines the next Error correction code to be used for protecting the remaining data. (Length 32 bits);

**EPB data:** contains the data enabling the correction

(typically redundancy bits) for the chosen RS code.

It is to be noted that the error-correcting code settings (*i.e.* the definition of the code protecting the remaining marker segments) can be specified in the beginning of the EPB marker segment in the Pepb parameter. The only restriction is that , these parameters have then to be of fixed size, as they will have to be protected by the first (fixed) code.

For efficiency reasons, it is necessary to consider that there could be more than only one EPB in the header. In fact, the Main Header or Tile-part Header size can be quite large when optional markers such as PPM (packed packet headers) are included. As a consequence, it is useful to include within the EPB syntax an EPB index (in Depb) which will enable the presence of several EPBs in the header. By default, one could consider that if the index is set to 0, the EPB block is the only one present, otherwise the EPBs are grouped together.

Another element to be considered is the large variations of the mandatory fields size, in particular in the Main Header. As a matter of fact, the number of components in the image may vary very much, even though most of the images used have in practice up to three or four components. Considering that this number of components directly impacts the size of field SIZ, which itself must be protected by the first fixed code, the dimensioning of the first code to the maximal possible size of SIZ field would lead to dramatic compression efficiency losses. As a consequence, it is useful to consider that by default the EPB will be dimensioned to match the most common cases, typically up to three components. Should the image contain more components, a second or more EPB marker placed immediately after the first one will protect the rest of the mandatory fields (typically the end of SIZ). At the decoding side, an hypothesis test on the number of image components will be carried out using the expected associated redundancy. The decoded number of components is the one with the lowest error detection test.

For more detailed information on the proposed EPB, please refer to [6].

#### 2.5. Default error correction codes

A default error correction code has been defined to be used in the simulation tests for protecting data preceding EPB redundant data. As previously mentioned, it is mandatory to define this default code for the two following reasons:

- the EPB marker segment cannot define the error correction code to be used for itself;
- the position of the first EPB marker segments

included in the Main Header is not known precisely and depends of the number of component of the image, which impact the size of the SIZ marker segment. This constraint imposes to perform first a synchronisation based upon the error detection process.

When the number of data to be protected by RS(N,K) is not a multiple of K bytes, it is necessary to generate padding bytes. In order to limit this byte padding, it is interesting to chose codes astutely, namely codes leading to a total size  $L1+L2$  in bits divisible by 8. Considering that one and three image components are the most common case, and considering the proposed EPB structure, we selected as default code the code already proposed in [8] for the main header, *i.e.* the Reed-Solomon code RS(128,64). This code allows to correct up to 32 erroneous bytes. Note however that the use of a BSC channel, which means that the errors are uniform across the codestream, is not optimal for the Reed-Solomon code, as when the error bits are distributed in different bytes, they may lead to an error correction limit of only 32 bits (in the worst case). Distributed in the 128 bytes (1024 bits) of the EPB for the main header, those 32 error bits account for a BER of  $3.10^{-2}$ , which is clearly our error correction capacity, as can be seen in the results presented in Section 3.

In order to limit byte padding two other default codes have been used:

- RS(45,25) for the first EPB marker segment of a Tile-part Header;
- RS(25,13) for the non-first EPB marker segments of both the Main Header and the Tile-part header.

### 3. RESULTS

In this section, some results about above described protection scheme are provided. In a first step, compliance tests are carried out, to address the problem of the backward compatibility. Then, tests on the robustness to errors of the detection of the number of image components are proposed. Finally, a comparison with the case without header protection is done in presence of noise, and the number of decoding leading to a decoder crash are counted.

#### 3.1. Backward compatibility compliance tests

To ensure that the insertion of the EPB marker segments described in Section 2 allows as foreseen to keep backward compatibility with JPEG 2000 Part 1 decoder, compliance bitstreams (JPEG 2000 Part 4) embedding the EPB marker segments have been generated and tested

with the different JPEG 2000 reference software. Those bitstreams, which are available for testing on the JPEG 2000 member's website (<http://www.jpeg.org>), were fully decoded and provided the same output as their counterparts without EPB marker segments.

#### 3.2. Synchronization tests

As mentioned before, JPEG 2000 allows to work with multi-component images. As a result, the size of the headers to be protected increases with the number of components, and it was chosen to perform as first step before error correction a detection of the number of components of the image. This detection can be done by considering a hypothesis test on the number of image components (typically 1 or 3 which are the most likely figures in practice) using the expected associated redundancy. The more likely number of components is the one with the lowest error detection test. This detection can be viewed as synchronisation test, as we try to recover the start of redundancy in the compressed bitstream.

Tests have been done to determine the robustness of such a mechanism in presence of errors. The case of a Binary Symmetric Channel (BSC) with various resultant Bit Error rates (BER) was considered. For a given image and a given number of components, the percentage of good detections of the number of components has been estimated.

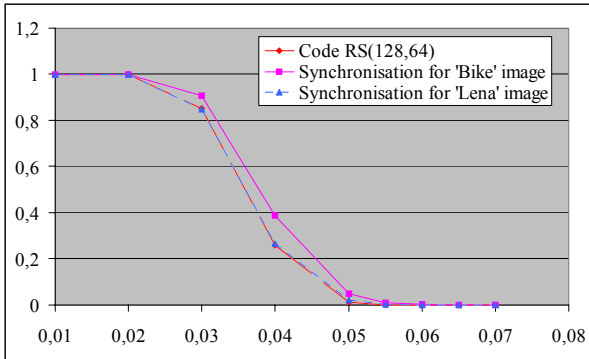
Table I shows the results obtained for image "Bike" (2048x2560, 1 component) and image "Lena" (512x512, 3 components), both compressed at 0.5 bpp (bit per pixel) for various BERs. The probabilities of good detection were derived for 10000 independent noise realisations. For each realisation, the channel bit error rate constant through the image, which means that the compressed bitstream (headers and data) is uniformly corrupted.

**TABLE I**  
PROBABILITY OF GOOD DETECTION AVOIDING A DECODING CRASH FOR ONE AND THREE COMPONENTS IMAGES CORRUPTED BY A BSC CHANNEL.

BER on the BSC	Image "Bike"	Image "Lena"
$10^{-3}$	1.0	1.0
$10^{-2}$	1.0	1.0
$2.10^{-2}$	0.99995	0.999
$3.10^{-2}$	0.908	0.850
$4.10^{-2}$	0.389	0.267
$5.10^{-2}$	0.049	0.022
$6.10^{-2}$	0.002	0.001
$7.10^{-2}$	0.0	0.0
$10^{-1}$	0.0	0.0

It can be seen that a highly reliable detection of the

number of image components is obtained for a large range of BER. The error correction capabilities depend from the considered RS codes capabilities. This is illustrated in Fig. 5, where the obtained synchronisation results are compared with the considered RS(128,64) code performance. As foreseen due to the use of padding for images with a number of components different of 3, results for 'Lena' image are the same as the RS code ones, whereas those obtained for 'Bike' image are slightly better. Naturally, better results (a larger BER range where good detection are obtained) would be obtained by taking



a more powerful code.

Fig. 5. Performance comparison for synchronisation tests and standard RS decoding.

### 3.3. Crash decoding tests

Having tested the performance of the synchronisation process, we now consider the overall performance of the proposed system by tracking performance in terms of crash decoding. Errors occurring in header may indeed conduct to a crash of the decoding process, or to place it in a never-end state. As the focus of this paper is the study of error detection and correction mechanisms for the main and tile-part headers, our purpose is to show that such crashes in the decoding process can be avoided using the proposed header protection mechanism. Note that when no decoder crash occurs, *i.e.* when an output image is produced by the decoder, this image may still result from a codestream that include errors in the headers. In this case, the decoded image may or may not be damaged seriously, depending on the location of the error(s). For example an error occurring in an optional marker segment may have no impact on the decoded image, whereas an error in the image size will conduct to a bad result. In this section we consider only the fact that images can be decoded without freezing or crashing the decoder, whatever the result. It is to be noted that the crash behaviour strongly depends of the decoder implementation, and that given performances are only valid with the decoder used in the experiments, *i.e.* the

**TABLE II**  
PROBABILITY OF AVOIDING A DECODING CRASH FOR "LENA"  
IMAGE CORRUPTED BY A BSC CHANNEL.

BER on the BSC	Image "Lena"
$10^{-4}$	0.94
$10^{-3}$	0.63
$10^{-2}$	0.03
$2 \cdot 10^{-2}$	0.0
$3 \cdot 10^{-2}$	0.0
$4 \cdot 10^{-2}$	0.0
$5 \cdot 10^{-2}$	0.0
$10^{-1}$	0.0

Verification Model version 8.6.

Once again, a BSC channel was considered for our experiments. The corresponding results obtained for both the mono and multi tile-part cases with the mechanism defined in Section 2 are compared to those obtained without using backward compatible error correction capabilities.

As reference, Table II shows the results obtained for the "Lena" image compressed at 0.5 bpp, with one tile and one tile-part, when no EPB marker is considered. The probability of crash avoidance was derived for 1000 independent noise realisations. Those results illustrate the dire need for header protection, as it is obvious that even for a BER as low as  $10^{-4}$ , the decoder may crash, leading to an interruption of the service, the obligation to re-initialise the decoder ("reboot"), .... Let now compare these results to those obtained for the same image (still compressed 0,5 bpp) with the EPB marker embedded.

**TABLE III**  
PROBABILITY OF AVOIDING A DECODING CRASH WITH THE EPB  
MARKER SEGMENT FOR "LENA" IMAGE CORRUPTED BY A BSC  
CHANNEL, IN THE MONO OR MULTI TILE-PART CASE.

BER on the BSC	Image "Lena" 1 tile-part	Image "Lena" 5 tile-parts
$10^{-4}$	1.0	1.0
$10^{-3}$	1.0	1.0
$10^{-2}$	1.0	1.0
$2 \cdot 10^{-2}$	0.966	0.965
$3 \cdot 10^{-2}$	0.672	0.567
$4 \cdot 10^{-2}$	0.134	0.036
$5 \cdot 10^{-2}$	0.003	0.0
$10^{-1}$	0.0	0.0

Table III shows the results obtained firstly in the case where a unique tile-part is used and secondly in the case where the image is coded with five tile-parts. Once again, the probabilities of crash avoidance were obtained for 1000 independent noise realisations.

The results obtained in those various tables clearly

show the interest of the proposed EPB mechanism, for it provides a clear improvement of the decoder performance in presence of transmission errors in the JPEG 2000 headers. Typically, for a bit error rate of  $2 \cdot 10^{-2}$  on the channel, almost all images are decodable (in more than 96% of the cases) whereas the classical image systematically leads to a decoding crash. This is even more interesting those results are obtained both for the case of mono and multi tile-part, and for a very reduced overhead cost (about 0.35% for "Lena" images and about 0.11% for "Bike" image), which reflects from the authors point of view the flexibility and efficiency of the proposed header EPB marker segment technique.

#### 4. CONCLUSIONS AND PERSPECTIVES

A fully JPEG 2000 Part 1 compliant backward compatible error protection scheme for headers was proposed in this paper. The description of a new normative embedded marker segment was provided and tested with Reed-Solomon codes. Simulation tests were given, that show that the performance of the decoder can be significantly improved in terms of crash decoding for both mono or multi tile-parts, and with a reasonable decrease of compression efficiency.

As an immediate perspective of interest, using the particularly flexible structure of the EPB marker segment, embedded Unequal Protection schemes can be envisaged for joint or tandem Source-channel coding.

#### 5. REFERENCES

- [1] *JPEG 2000 image coding system*. ISO/IEC 15444-1 / IUT-T T.800.
- [2] L. Liang., B. Luc, A. Lie, J. Wus and F. Kossentini, "Error resilience Ad-hoc Sub-Group Report for the Tokyo Meeting", *ISO/IEC JTC 1/SC 29/WG 1*, N1606, Tokyo, Japan, Mar. 2000.
- [3] C. Poulliat and D. Nicholson, "Impact and efficiency of error resilience tools for mobile applications", *ISO/IEC JTC 1 SC29 WG1* N2176, Stockholm, Sweden, Jul. 2001.
- [4] A. Natu and D. Taubman "Unequal Protection of JPEG 2000 Code-Streams in Wireless Channels", *Proceedings of IEEE GLOBECOM'02*, vol. 1, pp. 534-538, Taipei, China, 17-21 Nov. 2002.
- [5] V. Sanchez and M.K. Mandal, "Robust transmission of JPEG 2000 images over noisy channels," *Proceedings of IEEE ICCE'02*, pp. 80-81, 2002.
- [6] D. Nicholson, C. Lamy, C. Poulliat and X. Naturel, "Backward Compatible Header Error Protection in a JPEG 2000 codestream", *ISO/IEC JTC 1 SC29 WG1* N2851, Seoul, Korea, Mar. 2003.
- [7] F.G. MacWilliams, N.J.A Sloane, *The Theory of error correcting codes: Part 1*, North-Holland Publishing Company: New York, 1977, pp. 294-315.

- [8] C. Poulliat, P. Vila, D. Pirez and I. Fijalkow, "Progressive quality JPEG 2000 image transmission over noisy channel", *Proceedings of EUSIPCO'02*, Toulouse, France, Sept. 2002.