Authentication and access control in the JPEG 2000 compressed domain

Raphaël Grosbois, Pierre Gerbelot and Touradj Ebrahimi

Signal Processing Laboratory - Swiss Federal Institute Of Technology, CH-1015, Lausanne, Switzerland

ABSTRACT

Transmission and exchange of digital images with friends and customers is become a very simple task thanks to the development of the communication networks and the tools built around them. Unfortunately, such operations become delicate whenever image security is required, typically for commercial applications or protection of proprietary data. Solutions associated to data encryption already exist but are usually complex and do not take into account the specificities of images, generally under a compressed form. The Joint Photographic Experts group has recently created a new still image coding standard called JPEG 2000. It presents an efficient compression scheme together with support of functionalities required by today and tomorrow applications (progressive decoding, region of interest...).

By considering the JPEG 2000 algorithm, we are presenting tools for image authentication or access control (on image resolutions and qualities). These techniques can be applied on JPEG 2000 codestreams or directly integrated into the coding/decoding operations, and are mainly based on modification and insertion of information in the bit stream. The resulting codestreams remain compliant with the standard. Moreover, they allow image side information retrieval and/or prevent use by unauthorized parties.

Keywords: JPEG 2000, image authentication, access control, compressed domain, wavelet domain, encryption, digital signature

INTRODUCTION

With the expansion of the communication networks, it appears relevant to develop tools to authenticate either the integrity of the data or the degree of control from an end-user. Some requirements are very important for applications where a digital image has to be sent to an end-user: First, the user should be able to verify the origin of the received data. Second, the sender generally wants to ensure that the data will not be used by an unauthorized party. Moreover, for efficient transmission and storage purposes, images are generally sent under their compressed form. A new still image compression standard, JPEG 2000, has been defined recently, to be used in various multimedia applications. This standard has been developed in order to integrate features needed by today and tomorrow applications (good quality at high compression ratios, scalability, Region of Interest,...), and that are generally either missing or not well covered in existing coding algorithms.[3]

This paper proposes one authentication and two access control techniques that can be easily integrated in a JPEG 2000 codec, while remaining compliant with the standard.

In the first section of this paper are detailed the requirements of the proposed techniques. Then, section 2 briefly describes the key points of the JPEG 2000 baseline algorithm. Section 3 presents the authentication algorithm. The two next sections describe the access control techniques on image resolutions and layers respectively. Finally, conclusions are drawn in the last section.

Further author information: (send correspondence to Raphaël Grosbois)
Raphaël Grosbois: E-mail: Raphael.Grosbois@epfl.ch
Pierre Gerbelot: Email: Pierre.Gerbelot@epfl.ch
Touradj Ebrahimi: E-mail: Touradj.Ebrahimi@epfl.ch

1. REQUIREMENTS

As mentioned above, the proposed techniques are designed to satisfy the needs of applications dealing with secure digital image transmission. More precisely, we are considering scenarios where a digital image has to be sent from its owner to an authorized end-user.

Such applications mostly make use of various encryption techniques, where an image is generally encrypted with a private key, and sent to an user that owns the public key necessary to decrypt it. These techniques are generally very robust, but with the cost of using complex algorithms. Likewise, they usually do not take into account the properties of the data and its encoding algorithm, hence preventing features such as random access or image progressive decoding.

The following requirements were identified and used during the techniques presented in this work:

1. The authentication should identify any attack modifying the values of the image coefficients.
2. The access control should allow partial visualization of the image content (i.e. low resolution or quality).
3. The access control should efficiently prevent unauthorized parties to access the protected data.
4. The proposed technique must remain compliant with the JPEG 2000 algorithm.
5. The proposed technique should preserve the image quality and the compression performance.
6. The proposed technique should be directly integrated in a JPEG 2000 codec without significant complexity overhead.

The first requirement implies that the authentication technique is should not be sensible to manipulations that preserve the values of the coefficients. For instance, transcodings that modify the progression mode of a JPEG 2000 codestream or that discard image high resolutions, must not be detected by the proposed algorithm. Requirements 2, 3 and 4 signify that all users should be able to decode the compressed image, but only authorized one can access to the original data. Finally, requirements 5 and 6 indicate that the proposed techniques can be applied during the image encoding and/or decoding procedures (either in the wavelet domain or on the bit stream), while not decreasing the performances and the features of the codec.

2. JPEG 2000 OVERVIEW

The JPEG 2000 standard has been created by the Joint Photographic Experts Group (JPEG), also denominated as ISO/IEC JTC1 SC29/WG1. Currently, it defines seven parts dealing with the different aspects of the standard. Although at the time of the writing of this paper, only the first part has reached the level of International Standard (IS), the currently defined parts are:

- Part 1: Baseline mode (core system).[1]
- Part 2: Extensions, covering specific applications.[2]
- Part 3: Motion JPEG 2000.
- Part 4: Conformance and compliance.
- Part 5: Reference software.
- Part 6: Compound images file format.
- Part 7 (called part 8 in the specification): JPEG 2000 hardware reference code.

In this paper, we are essentially concerned by the first part of the standard, although the proposed techniques are likely to be extended to the other parts. JPEG 2000 part 1 only defines the decoding algorithm as well as the codestream syntax. Nevertheless and for the purpose of a clear presentation, we describe a generic encoding algorithm in this section. More information and details can be found in the description of the standard.[1]

A typical JPEG 2000 part 1 encoding algorithm (see Figure 1) typically accomplishes four operations: Wavelet transform, embedded scalar quantization, entropy encoding and codestream building (i.e rate-allocation). The remainder of this section briefly describes each of these operations in order to facilitate the understanding of the authentication and access control methods. Note that some other modules, such as region of interest scaling,[4] are not represented here but can be present in more sophisticated encoders.

Figure 1. Typical JPEG 2000 encoder

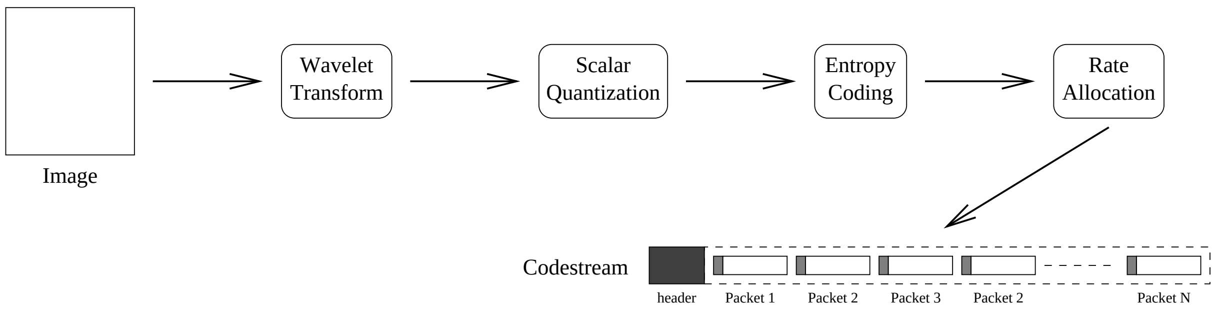
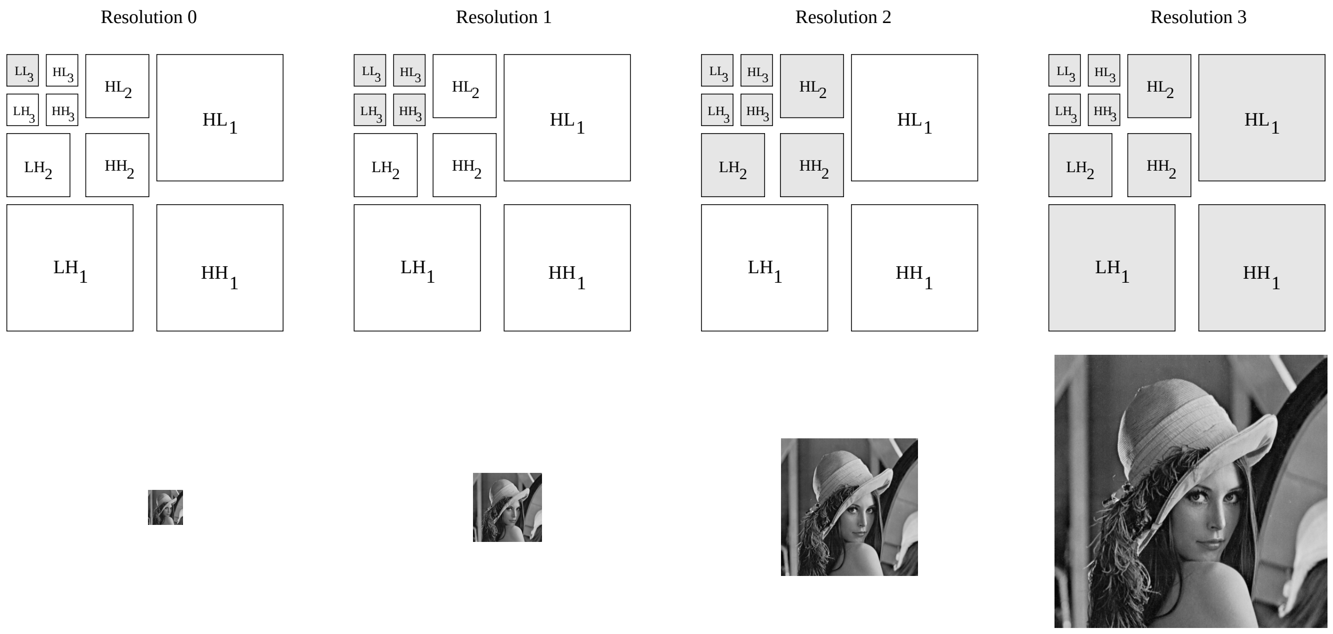


Figure 2. Image in the wavelet domain after three levels of decomposition. By using subbands information in the order LL_3, (HL_3,LH_3,HH_3), (HL_2,LH_2,HH_2) and (HL_1,LH_1,HH_1), it is possible to reconstruct resolutions 0,1,2 and 3 of the image respectively.



2.1. Wavelet Transform

During a first step, the image is decomposed into wavelet subbands, by applying a 1-D Discrete Wavelet Transform scheme (1-D DWT), successively on the rows and the columns.[5] JPEG 2000 part 1 actually supports two filter banks, corresponding both to biorthogonal wavelet bases. The first is based on Daubechies 9x7 wavelet filters[6] and is used for lossy compressions, because of its good performances at high compression ratios. The second is based on Le Gall 5x3 wavelet filters[7] and provides for lossless compressions thanks to its rational filters coefficients.

Thanks to this multi-resolution analysis, it is possible to access and work only in certain resolutions of an image. As depicted in Figure 2, the lowest image resolution corresponds to the LL wavelet suband (Low horizontal and Low vertical frequencies). The second image resolution is reconstructed by adding the information of the HL (High horizontal and Low vertical frequencies), LH (Low horizontal and High vertical frequencies) and HH (High horizontal and High vertical frequencies) subbands of the last decomposition level. Likewise, one reconstructs the highest image resolution from the last but one resolution and the HL, LH and HH subbands of the first decomposition level.

Figure 3. Sign-magnitude representation of a coefficient. The sign is located in the Most Significant Bit (MSB) and directly followed by the absolute value of the coefficient's magnitude.

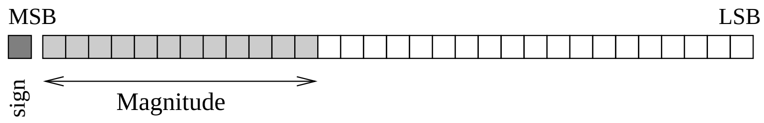
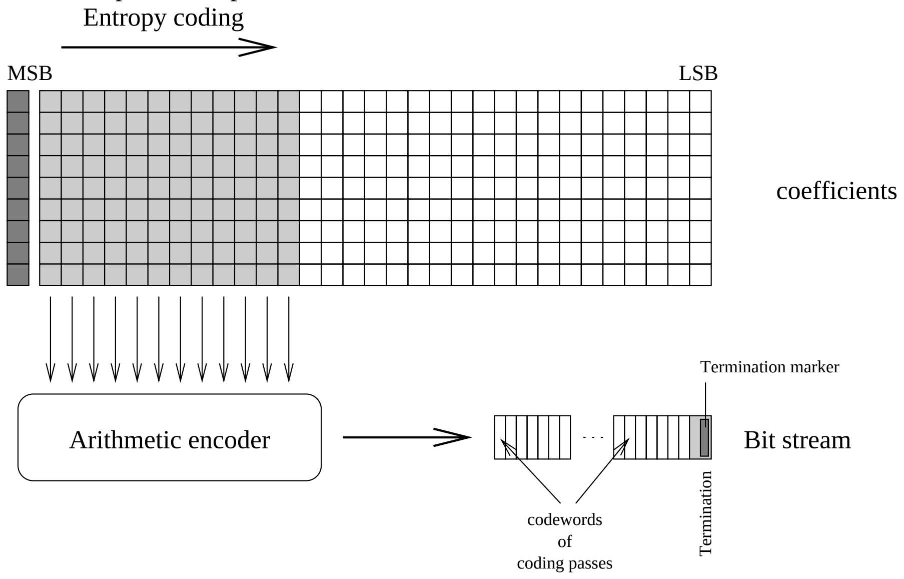


Figure 4. Arithmetic encoding of code-block's bit-planes. The bit-planes are processed in a maximum of three coding-passes that generate codewords. The resulting bit stream must be terminated, at least, after processing the last bit-plane. The last termination marker is either implicit or explicit



2.2. Embedded Scalar Quantization

The second step in a JPEG 2000 compression is a scalar quantization of the wavelet coefficients according the formula:

$$w_q[s,i,j] = sign(w[s,i,j]) * \lfloor \frac{|w[s,i,j]|}{\Delta_s} \rfloor \tag{1}$$

where $w[s,i,j]$ and $w_q[s,i,j]$ are the un-quantized and quantized wavelet coefficients at location (i,j) in subband s, Δ_s is the quantization step size of subband s, $sign(x)$ gives the sign of coefficient x, and $\lfloor x \rfloor$ is the floor function (i.e. largest integer not exceeding x).

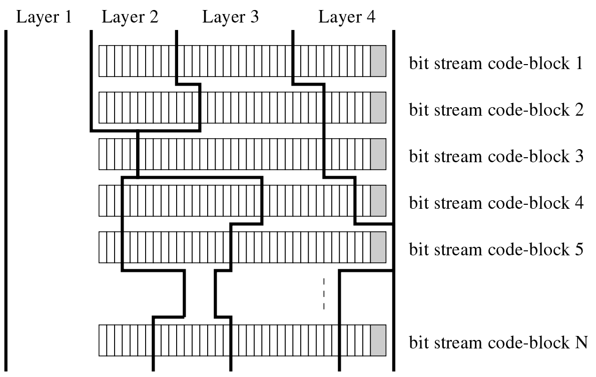
This operation reduces the precision on the coefficients and is obviously not applied in the case of a lossless compression. Nevertheless, it is worth pointing out that, from this step, all the coefficients must be used in their sign-magnitude representation, as sketched in Figure 3. Note that in this representation, the magnitude of the coefficients is shifted upward the Most Significant Bits (MSB's), and may be followed by fractional bits useful to either estimate the distortion during the rate-allocation process[8] or to scale coefficients in ROI scaling mode.[4]

2.3. Entropy Coding

The entropy coding is performed inside rectangular groups of wavelet coefficients called *code-blocks*. In each code-block, the coefficients in sign-magnitude representation are processed from the most significant magnitude bit-plane toward the least significant. Furthermore, in each bit-plane the bits are scanned in a maximum of three passes called *coding-passes*. Finally, during each coding-pass, the scanned bits with their context value are sent to an adaptive arithmetic encoder that generates the code-block's bit stream. The whole operation is summarized in Figure4. More details can be found in the description of JPEG 2000 part 1.[1]

It is important to note that the codewords generated by the arithmetic encoder must be (at least) terminated at the end of the last bit-plane (the standard actually allows terminations after any coding-pass). During this termination procedure, the content

Figure 5. Layers building from code-blocks coding-passes codewords



of the arithmetic encoder's registers is flushed into the bit stream, which is then ended by an explicit or an implicit termination marker (i.e. a two-bytes value greater than 0xFF8F). In the latter case, no such marker is effectively written into the bit stream, based on the fact that the arithmetic decoder automatically adds such a marker when it reaches the end of the bit stream.

2.4. Codestream building

This last procedure carries out the creation of the so-called *layers*, as well as the generation of the final codestream. The JPEG 2000 layers roughly stand for successive qualities at which a compressed image can be optimally reconstructed. These layers are built during a rate-allocation process that collects, in each code-block, a certain number of coding-passes codewords (see Figure 5). Hence, in a code-block, the bit stream is distributed into a certain number of layers. Note that the last bytes of the bit stream may be discarded, resulting in a higher distortion for the reconstructed code-block. Furthermore, a code-block may not contribute to some layers. Details on the rate-allocation procedure can be found in D. Taubman's paper.[8]

A JPEG 2000 codestream (as shown in Figure 1) consists in a header followed by packets of data. In each packet appear the codewords of the code-blocks that belong to a same image component, image resolution and layer. Thus, a packet corresponds to a body, with the coding-passes codewords, and a header that contains identification and information about the corresponding code-blocks.

3. AUTHENTICATION

As pointed out earlier, our goal is to identify the attacks on image coefficients values. Since a modification in the spatial domain translates into a modification of wavelet coefficients, it turns out that, in the JPEG 2000 framework, the authentication procedure can be applied in the wavelet domain. Note that, due to the space-frequency localization of wavelet bases, it is still possible to determine the position of an eventual attack from the wavelet domain.

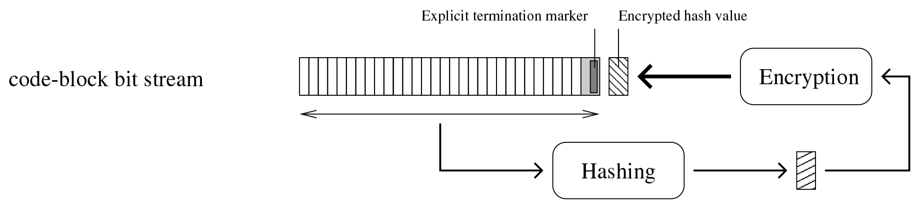
In our scheme, we propose an authentication of code-block's bit stream. Indeed, a modification of some wavelet coefficients in a code-block, generally implies the generation of different bit stream. Hence, we will consider as negligible any attack that does not modify the code-block's bit stream. Such an attack can be, for instance, a redistribution of the code-block's bit stream among the layers, or a modification of wavelet coefficients values that is less than the quantization step size.

From these considerations, we have designed a scheme that first computes a digital signature (i.e hash value) of the code-block's bit stream. The originality of the proposed method is in the inclusion of this hash value directly in the bit stream. More specifically, we use the fact that any byte in the bit stream, appearing behind a termination marker, will not be read by a compliant JPEG 2000 entropy decoder. Consequently, the proposed authentication technique corresponds to the three steps described below.

3.1. Code-block's bit stream hashing

In our method, we have used the Secure Hash Algorithm (SHA[9]) in order to generate digests for the code-blocks bit streams. Note that any other hash function (MD2, MD4, MD5, ...) could also be used in our authentication algorithm. The choice of such a function depends on the application's requirements, such as the difficulty to retrieve the bit stream corresponding to a hash value, the algorithm simplicity, or the influence on compression performance (because the hash value is inserted in the codestream). Thus, by applying the SHA algorithm on the bit streams, we generate digital signatures with a fixed length of 160 bits per code-block.

Figure 6. Code-block's bit stream authentication scheme



3.2. Hash value encryption

In a second step, we want to avoid that the hash value can be replaced after an attack, disabling thus any authentication. In this purpose, we encrypt each hash value with a public-key cryptosystem such as the Rivest, Shamir, and Adleman algorithm (RSA[10]). Again, any other cryptosystem can be used in the framework of our method, depending on the requirements of the application. In any case, at the end of this operation, we end with the encrypted hash values of all code-blocks bit streams.

3.3. Encrypted hash value insertion in the bit stream

By construction, the arithmetic decoder stops reading bytes from the bit stream when it encounters a termination marker (i.e. a two bytes value greater than 0xFF8F). This interesting feature is used here to add some extra bytes in the bit stream without affecting the arithmetic decoding process. Indeed, any added bytes will be skipped by JPEG 2000 compliant decoders. The only precaution to take is the number of added bytes since this increases the codestream bit-rate.

In our method, we append the encrypted hash value at the end of the corresponding bit stream. However, because of possible use of implicit termination markers, we have first to ensure that the last termination marker is effectively written in the bit stream, before adding the encrypted signature. Note that thanks to the position of the hash value in the codestream, any JPEG 2000 compliant decoder (not aware of the authentication method), will skip the value and will perfectly decode the compressed image. This is why the proposed method claims JPEG 2000 compliance.

3.4. General considerations

The global authentication system is summarized in Figure 6. This operation is done between the entropy coding and rate-allocation procedures of a JPEG 2000 compression. At the decoder side, we retrieve first the code-block's bit stream, and isolate the bytes before the termination marker (i.e. coding-passes codewords) from those after this marker (i.e encrypted hash value). Finally, we recompute the new hash value of the codewords and compare it with the decrypted one. Note that an attack is detected when the encrypted hash value is missing or when it is not the one generated from the preceding bytes in the bit stream.

Although the JPEG 2000 standard allows several terminations inside a same code-block bit stream, only the last is used in our scheme. This is to limit the codestream bit-rate overhead (160 bits per code-block and per termination) and also to avoid being sensitive to modifications of the code-blocks contributions to layers (i.e. layer re-allocation).

As pointed out earlier, the originality of the proposed authentication algorithm resides in the place where the encrypted hash value is put. Thus, any other hashing and encryption algorithms can be used at the place of the proposed SHA and RSA techniques.

Finally, it is worth pointing out that the idea of inserting (hiding) some extra bytes after codeword termination can be extended to may other applications. The two next sections describe applications of this idea in the framework of image access control.

4. ACCESS CONTROL ON RESOLUTIONS

In this section, we present a method that allows a preview of low resolution(s) of an image, whilst preventing the correct display of its higher resolutions. The idea is to introduce a pseudo-random noise in the high frequencies subbands (i.e. scrambling of high resolutions), such that the decoded image appears very distorted for a decoder that does not know how to remove this noise.

Figure 7. Algorithm for image access control by resolution scrambling.

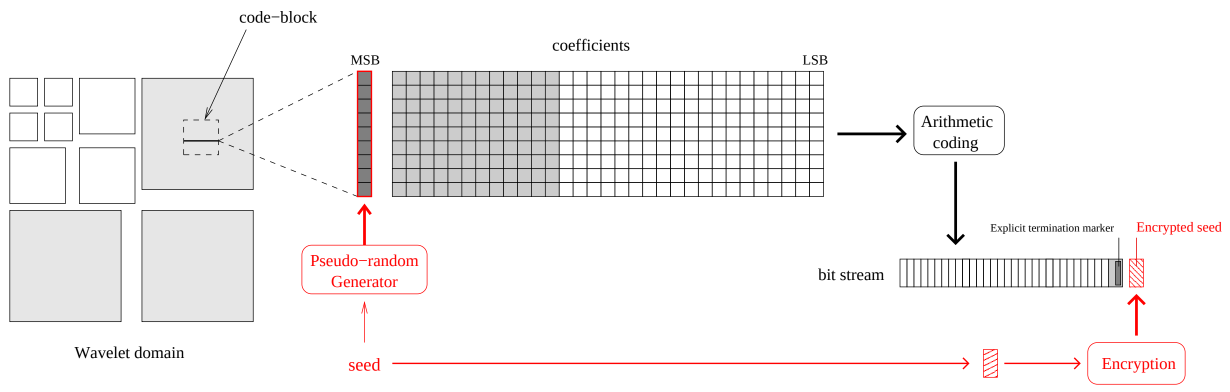
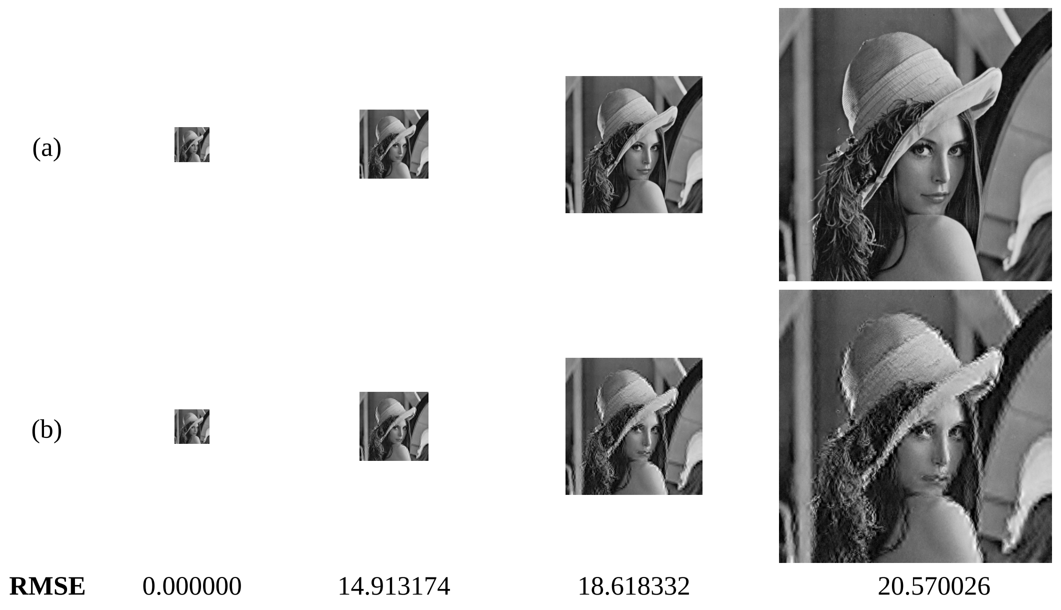


Figure 8. Resolution progressive decoding of a codestream which has been scrambled starting from resolution 1. (a) Authorized user (b) Unauthorized user.



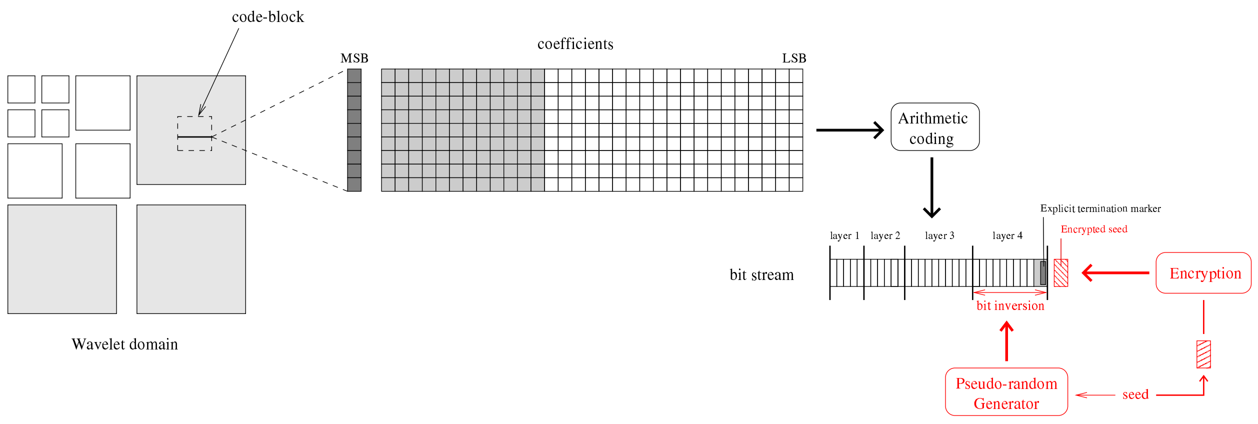
4.1. Pseudo-random inversion of coefficients signs

To *scramble* the high resolutions of the image, we add a known noise to the wavelet coefficients that belong to the high frequencies subbands (i.e the high resolution levels). In order to have a very simple algorithm, we inverse pseudo-randomly the signs of the coefficients in each code-block as depicted in Figure 7. Note that this method modifies only the most significant bit-plane of the coefficients and can be performed on-the-fly during the entropy coding.

In our experiments, we have used a pseudo-random generator based on a linear congruential formula[11] and characterized by its *seed* value. Thus, for each coefficient, a new pseudo-random value is generated and compared with a density threshold. If the pseudo-random value is greater than the threshold, the sign is inverted, if not the sign is left unchanged.

This operation is repeated with all coefficients (scanned in raster order) and in order to improve the security of this scrambling technique, we used a different seed in each code-block. Then, to communicate these seeds to the authorized users, we encrypted them, and put the encrypted values at the end of the corresponding bit streams (same method as in section 3 for the hash value).

Figure 9. Algorithm for image access control by layer scrambling.



4.2. Results

Figure 8 sketches the results obtained by scrambling resolutions 1 to 3 (resolution 0 is left intact) of an image that is compressed with three wavelet decomposition levels. We observe the qualities of the decoded images when disposing of the seed or not. Naturally in both cases, the first resolution is identically reconstructed, because no noise has been introduced. However, for higher resolutions, the error increases between the two corresponding image resolutions and the distortion obtained without the seed becomes increasingly visible.

A typical application for this method would be the browsing of a database of JPEG 2000 images where a certain number of high resolutions would have been scrambled. There, the client could select an image, download it, but would not be able to correctly display it before getting the public key of the encrytion technique necessary to retrieve the random generator's seeds.

Finally it is worth pointing out that the same information (i.e. codestream) is sent to all users, authorized or not. Consequently, the security of image is mainly ensured by the encryption technique used for the seeds.

5. ACCESS CONTROL ON LAYERS

This section provides for an access control on image qualities. Here, the idea is to introduce the pseudo-random noise directly in some layers of the image. The advantage of this method is that it can be applied directly on the bit stream (and not in the wavelet domain). Finally with this method, an unauthorized party will be able to display a rough quality of the image but this image will contain too much distortion to be printed or further used.

5.1. Scrambling of last layers

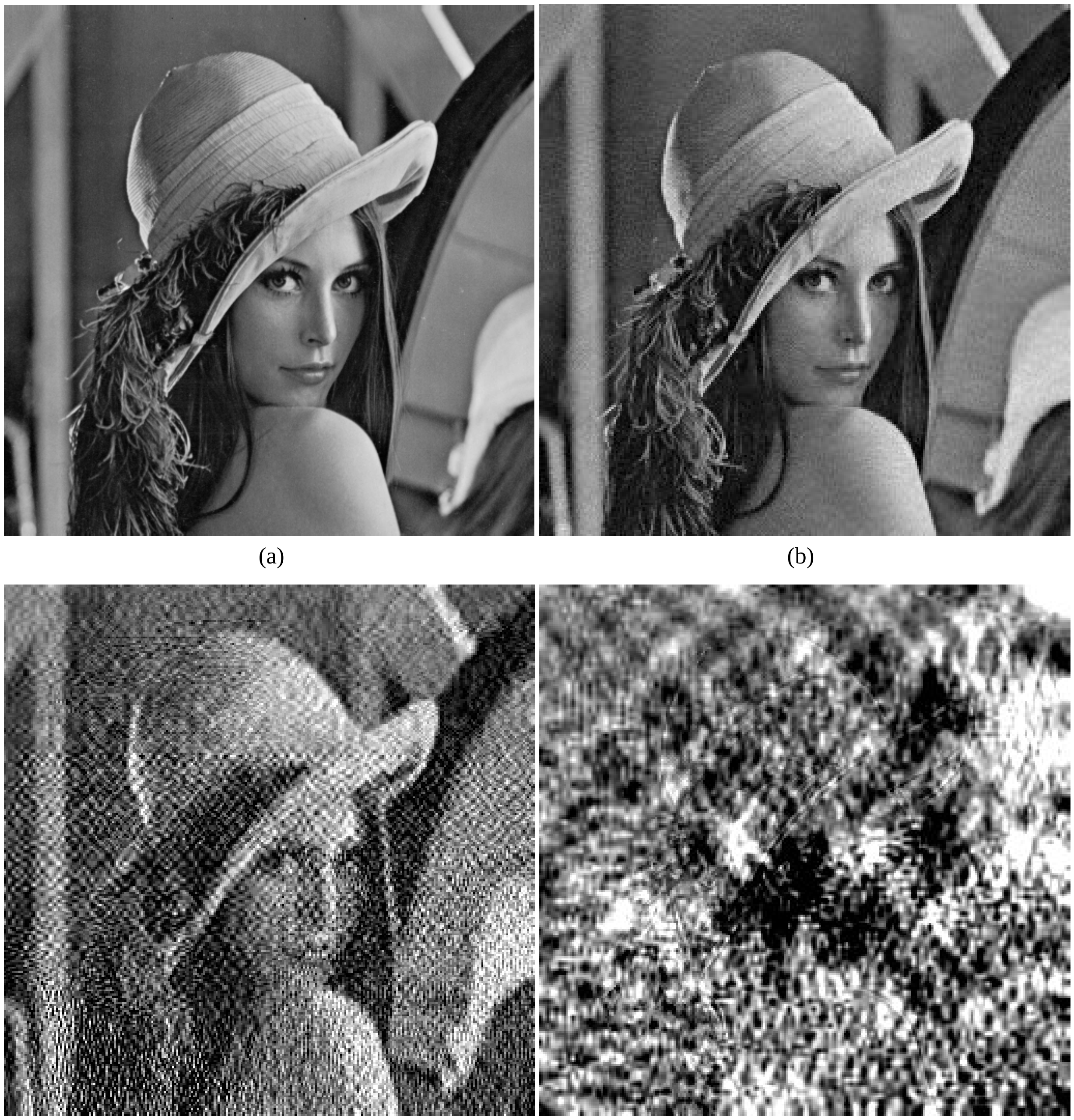
Using the same method as for image resolution scrambling, we pseudo-randomly invert the bits in the coding-passes codewords belonging to the last layers. For this, we use the same kind of pseudo-random generator identified by its seed and we put the encrypted version of the seed at the end of the bit stream as depicted in Figure 9.

5.2. Results

Figure 10 illustrate the application of the proposed method on a codestream that contains three layers. We observe the visual quality of the decoded images when the scrambling method has been applied on the last layer, on the two last layers and on every layer. As expected, the visual quality decreases with the number of scrambled layers. Note that in practice, we have also to indicate to the decoder how many layers have been scrambled by our technique. Furthermore, we can imagine applications where the number of scrambled layers differs from one code-block to another, and consequently apply a finer control on the distortion introduced in the final image (when decoding without knowing the seeds). Likewise, as layers following scrambled ones will not be correctly decoded, even if they are not scrambled itself (because of the arithmetic decoding), we can also scramble intermediate layers and left intact the first and last ones.

Finally, as for the previous method, it is worth pointing out that every user (authorized or not) receives the same codestream and can retrieve the original data as soon as a public key for decryption is provided.

Figure 10. Layer scrambling of a codestream where three layers have been defined. Decoding by an unauthorized party (a) Original image. (b) Third layer is scrambled. (c) Second and third layer are scrambled (d) All the layers are scrambled



CONCLUSIONS

In this paper, we have presented tools for authentication and image access control. These are all based on the fact that, by construction, the JPEG 2000 codestream allows insertion (hiding) of information after each coding-pass termination. Consequently, in the framework of image authentication, one can hide an encrypted version of each code-block bit stream's hash value. Likewise, by combining this technique with an image resolution scrambling or with a layer scrambling process, we have shown that it is possible to easily build an efficient tool for image access control. Finally it has been shown that all these techniques can be directly integrated in JPEG 2000 encoders and decoders, without breaking the compliance with the standard.

REFERENCES

1. *JPEG 2000 part 1 Final Draft International Standard*, ISO/IEC JTC1/SC29 WG1 N1890R, August 2000.
2. *JPEG 2000 part 2 Final Committee Draft*, ISO/IEC JTC1/SC20 WG1 N2000, December 2000.
3. D. Santa-Cruz, T. Ebrahimi, J. Askelöf, M. Larsson and C. A. Christopoulos, *JPEG 2000 still image coding versus other standards*, Proceedings of the SPIE's 45th annual meeting, Applications of Digital Image Processing XXIII , vol. 4115, pages 446-454, San Diego, California, Jul. 30-Aug. 4, 2000
4. C. A. Christopoulos, J. Askelöf, M. Larsson, *Efficient methods for Encoding Regions Of Interest in the upcoming JPEG 2000 still image coding standard*, IEEE Signal Processing Letters, Vol. 7, No. 9, Pages: 247-249, September 2000.
5. Stéphane Mallat, *A wavelet tour of signal processing*, Academic Press, 1998.
6. I. Daubechies, *Ten Lectures on Wavelets*, SIAM, Philadelphia, PA, 1992.
7. D. Le Gall and A. Tabatabai, *Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques*, IEEE International Conference on Acoustic, Speech and Signal Processing, Vol. 2, New York, pp. 761-765, April 1988.
8. D. Taubman, *High Performance Scalable Image Compression with EBCOT*, IEEE Transaction on Image Processing, Vol. 9, No. 7, Pages: 1158-1170, July 2000.
9. FIPS Publication 180, *Secure Hash Standard (SHS)*, NIST, May 11, 1993.
10. R.L. Rivest, A. Shamir, and L.M. Adleman,*A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM (2) 21 (1978), 120-126.
11. Donald Knuth, *The Art of Computer Programming*, Volume 2, Section 3.2.1.