Introducing keytagging, a novel technique for the protection of medical image-based tests

Óscar J. Rubio[a,*], Álvaro Alesanco[a], José García[a]

[a] *eHealthZ Research Group, Communications Networks and Information Technologies for E-health and Quality of experience group (CeNITEQ), Aragón Institute of Engineering Research (University of Zaragoza).*
Edif. Ada Byron, C/María de Luna 3, 50018 Zaragoza (Spain).

Abstract

This paper introduces keytagging, a novel technique to protect medical image-based tests by implementing image authentication, integrity control and location of tampered areas, private captioning with role-based access control, traceability and copyright protection. It relies on the association of tags (binary data strings) to stable, semistable or volatile features of the image, whose access keys (called keytags) depend on both the image and the tag content. Unlike watermarking, this technique can associate information to the most stable features of the image without distortion. Thus, this method preserves the clinical content of the image without the need for assessment, prevents eavesdropping and collusion attacks, and obtains a substantial capacity-robustness tradeoff with simple operations. The evaluation of this technique, involving images of different sizes from various acquisition modalities and image modifications that are typical in the medical context, demonstrates that all the aforementioned security measures can be implemented simultaneously and that the algorithm presents good scalability. In addition to this, keytags can be protected with standard Cryptographic Message Syntax and the keytagging process can be easily combined with JPEG2000 compression since both share the same wavelet transform. This reduces the delays for associating keytags and retrieving the corresponding tags to implement the aforementioned measures to only $\simeq 30$ and $\simeq 90\,ms$ respectively. As a result, keytags can be seamlessly integrated within DICOM, reducing delays and bandwidth when the image test is updated and shared in secure architectures where different users cooperate, e.g. physicians who interpret the test, clinicians caring for the patient and researchers.

Keywords: DICOM, Distortion, Keytagging, Privacy, Security

---

[*] Phone number: (+34) 976762698, fax number: (+34) 976762111

Email addresses: `orubio@unizar.es` (Óscar J. Rubio), `alesanco@unizar.es` (Álvaro Alesanco), `jogarmo@unizar.es` (José García)

Preprint submitted to Journal of Biomedical Informatics *May 2, 2015*

1. Introduction

The landscape of healthcare has evolved dramatically in the last few years with the digitization of medical image-based tests and the deployment of standard protocols for their storage and transmission, such as the Digital Imaging and COmmunication in Medicine (DICOM [1]) standard. The new e-health model [2] fosters ubiquitous and pervasive access to users of medical information: physicians who interpret the tests, clinicians caring for the patient, patients, researchers, medical teachers and students, etc. Furthermore, it enables the establishment of cooperative architectures where different authorised users may perform some editing of the image (e.g. adding annotations), update its attached information (e.g. examination-related data, patient demographics, health records) and share with others. On the one hand, this reduces the time for diagnosis and enables more accurate treatments, more thorough researches, and the provision of e-learning or commercial uses, depending on patient consent. On the other hand, the e-health model requires a high level of accessibility to medical information, which increases its potential risks. Since image-based tests include sensitive information, they are a target for eavesdropping, which may result in social and professional damage for the victim; and also for tampering and forgery, which may lead to misdiagnosis, mistreatments and wrong research outcomes. Current legal regulations such as the HIPAA [3] in the USA, the PIPEDA [4] in Canada and the LOPD [5] in Spain require the implementation of robust security policies to protect medical image-based tests. For this purpose, different security measures may be combined [6, 7]:

- Role-based access control (RBAC), so that each authorised user can read and/or edit certain contents of the image-based test according to his/her professional role, e.g. physician, researcher, teacher, etc.
- Integrity control, to detect if the image has been tampered with, which would endanger its clinical value.
- Tamper location, pinpointing the areas where the image has been manipulated, which may be helpful to validate images that are modified in permissible areas.
- Authentication, assessing if the image received corresponds to the image originally acquired, to an image derived from the original or to an unrelated image.
- Private captioning, associating private information with the image, only retrievable by authorised users of the authenticated image.
- Traceability control for user accountability, by associating marks from each entity that processes the image.

- Copyright protection, to pursue illegal copies if the image has a commercial use.

Hitherto, these security measures have traditionally been implemented in the medical context by means of cryptography, mainly in medical standards such as DICOM; or watermarking, mainly in research works. Nonetheless, these techniques present several disadvantages that limit their applicability for the protection of cooperative medical architectures (see Section 2). This paper proposes a new research direction, based on the goal of performing a secure and fast association of different types of data, called tags, to certain image features. These features and their associated tags may be categorized as stable, semistable or volatile, according to their robustness, i.e. their resistance to changing their values when the image undergoes some processing, such as compression or filtering. Stable tags would be suitable for authentication, traceability and copyright protection; semistable tags would be appropriate for private captioning with RBAC, and volatile tags would be ideal for integrity control and location of tampered areas. The association of tags to suitable features would be performed by means of a novel technique, called keytagging, whose main challenges are the extraction of image features with adequate robustness and the efficient encoding of the tags as a subset of the extracted features, by means of a map. This map will be compressed and protected to produce a stable, semistable or volatile keytag, which is an access key to retrieve the content of the tag from the image.

Compared to other alternatives, keytagging presents five relevant features, derived from the fact that this technique does not modify the medical image to associate information. First and most obvious, the image always preserves its clinical quality, without the need for assessment. Second, even very stable features can be used to encode keytags, which ensures high robustness and capacity. Third, no complex rules are necessary for the selection of the image features, since there is no risk of destroying its clinical value. Fourth, collusion and forgery attacks have no effect on keytagged image-based tests. Fifth, keytagging enables the deployment of secure and efficient cooperative architectures, since each user can add information related with the test by associating new keytags, with no risk of distorting or removing the previous ones. To share this update of the test, it is enough to send the new keytags to the rest of the users.

The rest of this paper is organized as follows. Section 2 analyzes related works and sets the background of this research. Section 3 depicts in detail the keytagging algorithms. Section 4 evaluates the performance of this approach by using different-size images from various medical acquisition modalities (CT, MRI, US, PET-CT), and image processing techniques that are common in the medical context. Based on the results retrieved, Section 5 proposes appropriate keytagging parameters for implementing the aforementioned security measures and depicts a risk assessment of the keytagging method. Next, Section 6 approaches the integration of keytags within the DICOM protocol, to enable the development of efficient architectures in which different users can update the DICOM file maintaining its security level. Finally, Section 7 outlines the main conclusions from this research and proposes future lines of work.

2. Related work

Keytagging has relevant influences from both cryptography and watermarking. The keytagging algorithm uses cryptographic tools to protect the keytags while the idea of associating data to different features of the image was motivated by watermarking techniques, which embed data into certain parts of the image. The advance introduced by keytagging is the ability to perform this association without embedding the data into the image, which permits avoiding any image distortion. Both keytagging and watermarking face a tradeoff between the robustness of keytags/watermarks and their capacity to host large contents, and both techniques require the protection of keytags/watermarks to prevent or hinder eavesdropping and forgery.

2.1. Cryptography

Cryptography provides two efficient and reliable tools to protect image-based tests, which may also be applied to watermarks and/or keytags. These are encryption, which may be symmetric or asymmetric; and hashing, which is found in digital signatures (DS) and hash message authentication codes (HMAC). The former can be used to implement RBAC by means of Cryptographic Message Syntax (CMS [8]), while the latter enables binary integrity control (tampered/non-tampered), authentication and traceability. Furthermore, private captioning can also be implemented by encrypting some content based on the hash of the image [9]. However, the location of tampered areas in an image would be burdensome since it would require multiple hashing of different parts of that image. The typical policy to protect an image-based test, e.g. a DICOM file, is to de-identify the private data of the image, include it with the information associated to the image, and use CMS to implement RBAC on this data, leaving the de-authenticated image unencrypted and adding a DS to the file to guarantee integrity and authentication. In addition, each entity that processes the file shall add its own DS for traceability.

The main issue in cryptography-based policies like this is the difficulty of developing cooperative architectures while maintaining the security measures. Any change in the test will invalidate all the previous signatures, and even though a new DS can be calculated, the traceability from the origin will be lost. An alternative is that each user adds his/her changes to the original test, digitally signs it and delivers the signed updated test to the rest of the users. In this way, the rest of the users can store the updated test with security, and add new updates from the last test version by following the same procedure. Nonetheless, as the number of users and updates of the medical image-based test grow, this approach becomes quite impractical in terms of delays, bandwidth and storage.

2.2. Watermarking

Medical Image Watermarking [10, 6, 7] (MIW) techniques embed limited amounts of hidden data (e.g. medical information), with one or several watermarks, within a medical image by means of image modifications and keys. The main feature of MIW is that the image can be manipulated (e.g. annotated, compressed with JPEG2000 [11], adjusted with different contrast and brightness) by the users without interference to its security regardless of whether they know about the existence of the watermarks. The keys used in watermarking, like those in symmetric encryption, are required to retrieve the embedded data and neither depend on the data nor on the image. There are different types of watermarks, and they may be combined to implement a security policy. Robust watermarks [12, 13], whose content is retrievable even if the image has undergone heavy modifications, may be used for authentication, traceability and copyright protection. Semifragile watermarks [14], whose content is retrievable only if those modifications are mild (e.g. if the image preserves its clinical value), may be used for private captioning. Finally, fragile watermarks [15, 16], whose content is retrievable only if the image is intact, may be used to implement integrity control and location of tampered areas in the image. Role-based control is also guaranteed since different watermarks, intended for different users, may be embedded in the same image. It is worth pointing out that MIW techniques require to produce a minimum distortion of the image to preserve its clinical value. In fact these techniques are often grouped in three categories depending on the manner how they cope with this requirement of high transparency.

The first group are the non-reversible techniques, which produce a permanent distortion on the image because they perform non-invertible operations, typically bit quantization, replacement or truncation. As a consequence, a thorough clinical assessment is necessary to guarantee that the clinical value of the watermarked images is preserved. Regarding watermarking domains, while the first approaches [17, 18] used the replacement of least significant bits (LSB) of pixels, which can only host fragile watermarks; the subsequent use of regions of different amounts of energy in transform domains allowed the embedding of multiple (robust, semifragile and fragile) watermarks in the same image [19, 20, 21]. Furthermore, medical data could be embedded in a compressed domain to combine the watermarking with lossy compression [22, 23].

The second group of MIW techniques corresponds to those that embed mainly in regions of non-interest ($RONI$) of the image. This minimizes the interference of the watermarks with the clinical content and avoids the need for clinical assessment. Robust watermarks are embedded in the $RONI$, surrounding the ROI [24, 25, 26] to try to avoid its deletion if the image is clipped, or in random locations [27, 28, 29] to increase the capacity. Only fragile watermarks for integrity control need to be embedded, at least partially, in the ROI. The security of these techniques against eavesdropping and forgery is low, since the modified pixels/coefficients where the watermarks were embedded are easy to identify in the $RONI$, which is usually black. Moreover, the $RONI$ may be used to insert visible watermarks or removed if medical image compression [30] is applied. In both cases the watermark/s would be partially or totally removed.

The third group are the reversible/lossless/invertible/erasable techniques, which distort the image but can recover its original quality by completely removing the watermarks after they have been detected and validated. This new concept of watermarking was first introduced in [31], and since then a variety of methods have been developed. Difference expansion, a kind of integer wavelet transform with high redundancy, was proposed early on in [32], obtaining low-distortion and high-capacity. Histogram operations, such as circular interpretation of bijective transformations [33], are also an effective manner to implement reversible watermarking with notable endurance to lossy compression. The addition of a virtual border where patient data is inserted in the LSB, at the cost of increasing the image size, has also been proposed [34]. The use of an estimator signal to determine which pixel blocks can embed information was proposed in [35], further used to embed a digest of the knowledge associated to the image [36], and refined in [37] by introducing a secure random location signal for security and implementing tamper detection and location. Similarly, an effective tamper location watermarking based on partitioning an authentication area into small regions in a multilevel hierarchical manner is depicted in [38]. Nevertheless, any reversible technique has two important drawbacks. First, it requires a secure environment since the image is unprotected once the watermarks are removed. Second, a user not allowed to access certain watermarks will not be able to remove them either, so he/she will work with a lower-quality version of the image.

It can be concluded that various disadvantages hinder the integration of watermark-based policies in standards such as DICOM and in cooperative architectures. First and most important, there is a tradeoff between capacity, robustness and image distortion. Thus, as new watermarks are added, especially if they are robust, the quality of the image decreases and the image may lose its clinical value, rendering it useless. Second, watermarks embedded by different users may interfere with each other since each new watermark may destroy part of the content of the others. Third, there is a tradeoff between robustness and capacity, so robust watermarks cannot be long. In addition to this, the use of watermarking in cooperative architectures would imply an important cost in bandwidth and delays, since every time that a user embeds a watermark in an image, he/she has to send the watermarked image together with the watermark keys to the rest of the users. If an image is watermarked several times by different users, it needs to be transmitted every time to the other users. In contrast, keytagging requires that each user has a copy of the images to be used for the keytagging. Once this requirement is met, different users can associate tags with the only further requirement that they transmit the corresponding keytags to the rest of the users.

3. Materials and methods

The procedure proposed for associating keytags $\{KT\}$, to bind the content of some tags $\{T\}$ to an image I_{or}, is formally described in Algorithm 1 and illustrated by means of an example in Figure 1; while the procedure for retrieving $\{T\}$ from I_{or}, or from some modified version $\tilde{I_{or}}$, is defined in Algorithm 2. $\{KeytagType\}$ is an important input parameter of these algorithms, which establishes the expected robustness of the tags when I_{or} undergoes common image modifications in the medical context and, according to it, their security applications (analyzed in Section 5). Stable tags are expected to be retrieved with low distortion even when $\tilde{I}_{or}$ has undergone aggressive image modifications which may have caused the loss of its clinical value, semistable tags are intended to remain undistorted only if $\tilde{I}_{or}$ has undergone mild modifications and preserves the clinical value of I_{or}, and volatile tags are intended to be retrieved highly distorted even if the modifications of $\tilde{I}_{or}$ are minor. Both Algorithm 1 and 2 rely on a preprocessing of the image, a selection of appropriate image features, a compact coding/decoding of the keytags and cryptographic protection of/access to the keytags. These processes are depicted in detail throughout Sections 3.1-3.4, which follow the notation described in Table 1. Finally, Section 3.5 describes how keytagging can be integrated within the JPEG2000 compressor.

3.1. Preprocessing

The first step of Algorithms 1-2 (line 2) is the segmentation of the region of interest (ROI) of the image, so that the tags can be associated to the most important parts of the image. The intention is to prevent the tags from being distorted or removed due to modifications affecting the $RONI$, such as medical image compression by areas [30], blackening private data for anonymization or the insertion of visible watermarks. If all the tag are associated outside the $RONI$, these modifications would be incapable of distorting or removing them. In some medical image modalities, an algorithm has been designed to obtain the ROI automatically, as is the case in [39] with the atherosclerotic plaque ultrasound. In addition to this, several medical image acquisition devices currently deliver the image ROI separated from the blocks of associated

Table 1: Operators and Notation

Notation	Meaning
$output/s \leftarrow f(input/s)$	Assignment of value to one or several outputs from a function or operator $f()$ with one or several inputs.
$[\ ]$	Concatenation operator.
$\lceil\ \rceil$	Operator of rounding to the nearest greater integer.
$\oplus$	Binary XOR operator.
$\{X\}$	Set of elements of type X, each element i represented as $X\{i\}$.
$\#X$	Number of elements that compose the set X.
$V(i:j:k)$	Vector derived from a vector V, corresponding to a subset of its elements, $[V(i), V(i+j), V(i+2\cdot j), ..., V(k)]$.
$M(:)$	Vector derived from a matrix M, corresponding to the concatenation of its rows, $[M(1,:), M(2,:), ..., M(end,:)]$.
I_{or}	Original image to be used for keytagging.
$\tilde{I_{or}}$	Modified version of I_{or} (e.g. compressed, filtered, clipped, rotated).
$ROI(I)$	Region of interest of an image I.
$RONI(I)$	Regions of non-interest of an image I.
MBR	Minimum bounding rectangle.
$R(I), G(I), B(I)$	Levels of R, G and B colors in RGB format of an image I.
I_g	Grayscale image derived from an image I.
CDF 9/7 or 5/3	Cohen-Daubechies-Feauveau 9/7-tap or 5/3 (also known as LeGall) filters.
$Coef, h, w \leftarrow WT(I, \underline{f}, j)$	Function that returns $Coef$, the j-th wavelet decomposition calculated with filters $\underline{\mathrm{f}}$ of an image I; h and w, the height and width of the wavelet decomposition levels from 1 to j.
$MWL(I)$	Maximum wavelet decomposition level of an image I.
LL	Coefficients of a wavelet subband obtained with horizontal and vertical low-pass filtering.
HL	Coefficients of a wavelet subband obtained with horizontal high-pass filtering and vertical low-pass filtering.
LH	Coefficients of a wavelet subband obtained with horizontal low-pass and vertical high-pass filtering.
HL	Coefficients of a wavelet subband obtained with horizontal and vertical high-pass filtering.
$aWL(I)$	Wavelet level allowed for an image I to associate certain tag/s.
C	Subset of coefficients from aWL.
F	Features from C used for the coding of one or several T.
$abs(X)$	Absolute value/s of X.
$LSB(X)$	Least significant bit/s of X.
T	Tag, binary data string to be associated to an I_{or} by means of a KT.
$\tilde{T}$	Tag retrieved from a modified image, $\tilde{I_{or}}$.
$s\_out \leftarrow LFSR(s, t)$	Linear feedback shift register with initial state s and taps t.
$GolombSeq$	A binary sequence that meets Golomb's randomness postulates.
$(X)^*$	A scrambled binary sequence derived from X by means of a reversible transformation.
BM	Bi-level map that encodes a tag as positions of certain coefficients of aWL.
KT	Keytag, which permits the retrieval of a T from an image.
Sk	Secret key used for symmetric encryption-decryption.
PrU	Private key to be used by user U for asymmetric decryption of data or for its signature.
PbU	Public key of user U, used by any user for asymmetric encryption of data intended for U, or to verify any signature issued by U.
$DS(D, \underline{Alg}, PrU)$	Digital signature of D using the algorithm Alg and the private key of the signatory U.
$checkDS(D, \underline{Alg}, PbU)$	Verification of the DS of D by using the algorithm Alg public key of the signatory U.
$encrypt(Plaintext, \underline{Alg}, K)$	Encryption of $Plaintext$ using the algorithm Alg and the key K.
$decrypt(Ciphertext, \underline{Alg}, K)$	Decryption of $Ciphertext$ using the algorithm Alg and the key K.

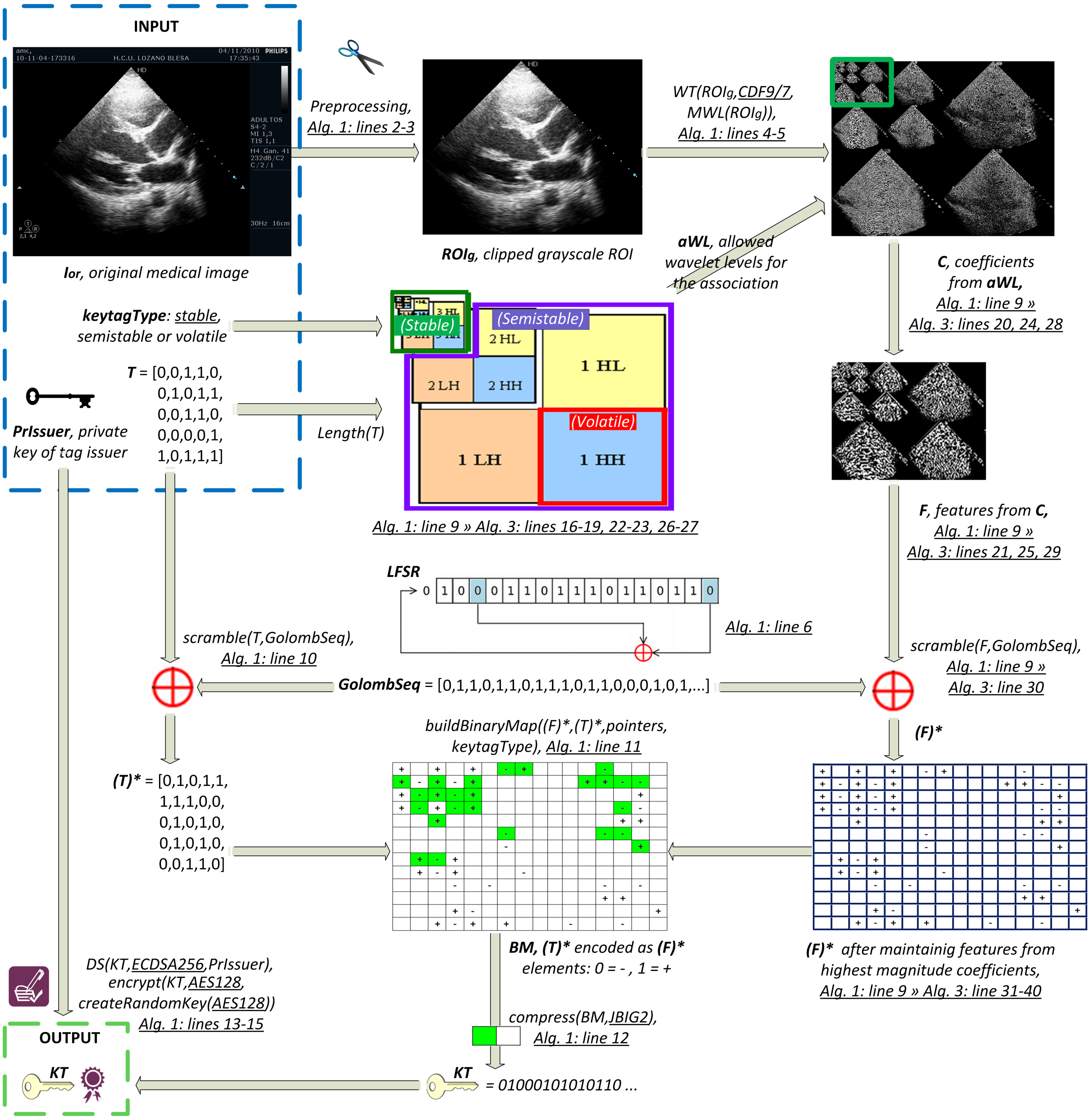

Figure 1: Main steps for the association of a stable keytag according to Algorithm 1.

data that compose the $RONI$. For example, in the DICOM standard [1] a calibration configuration was introduced for ultrasound images in which regions are defined with the same calibration. But since not all acquisition devices have this built-in capability, the ROI may be roughly segmented by means of some simple method, with the only conditions that the same method shall be used in both Algorithms 1-2 and that it shall leave out at least part of the black background and peripheral text of the image (if any). We recommend the automatic delineation of the minimum bounding rectangle (MBR) that encloses the medical image content (ROI). As a final step for the preprocessing, color ROIs are transformed into grayscale by calculation of their luma components according to the standard ITU-R Recommendation BT.601-7 [40] (Algorithms 1-2: line 3). This ensures compliance with both color and grayscale ROIs, and that further modifications of the color map does not affect the tag/s.

3.2. Selection of suitable image features

The 2-D Discrete Wavelet Transform [41] (Algorithms 1-2: line 5) decomposes the image into several scales, located in ordered regions of the transformed image, which host coefficients concentrating certain frequencies. This enables efficient compression (e.g. by means of SPIHT [42] or JPEG2000 [11]), since any entropy and/or run-length coding that exploits adequately the self-similarities of the quantized coefficients across different scales achieves high compression ratios; and also facilitates keytag association because the transform separates stable, semistable and volatile parts of the image. The main advantage of using wavelets over other transforms is its variable resolution: the higher frequencies, which correspond to details (volatile features of the image), are represented with higher spatial resolution than the lower frequencies. To obtain $WT(ROI_g, \underline{f}, j)$, ROI_g is initially filtered by rows and columns with two filters, decimated by two and arranged in four subimages: LL, LH, HL, HH. The process is iteratively repeated, taking the last LL as input, until reaching the desired j-th decomposition level. As a result, the lowest frequencies (most important parts, stable features) of the image are represented with only a few high-magnitude coefficients, located in the upper-left corner of $WT(ROI_g, \underline{f}, j)$ —note this in the 5th-level decomposition in Figure 1: upper right corner. The choice of the wavelet family, which sets the filters f, is relevant for compression but it was empirically found that it does not have a big impact on the robustness-capacity tradeoff. The only exception to the latter rule was observed when $\underline{f}$ are set to those used by a compressor, which improves the robustness to this compressor for a given capacity. Thus, our choice is using the filters implemented by the widespread JPEG2000 compressor [43] (see Section 3.5), CDF 9/7 for lossy compression and CDF 5/3 for lossless compression, which also saves a number of operations when keytagging is combined with compression (see Section 4.5). If the information about the compressor is not available, $\underline{f}$ is set to CDF 9/7

Table 2: Average energy and maximum number of coefficients in the wavelet subbands of the medical images from the test set.

Subband	Level 1		Level 2		Level 3		Level 4		Level 5		Level 6		Level 7		Level 8		Level 9	
	Energy	#Coefs	Energy	#Coefs	Energy	#Coefs	Energy	#Coefs	Energy	#Coefs	Energy	#Coefs	Energy	#Coefs	Energy	#Coefs	Energy	#Coefs
Approximation (LL)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	30.43	1
Horizontal detail (HL)	1.08	65,536	1.86	16,384	2.19	4,096	2.80	1,024	3.66	256	5.09	64	6.66	16	9.21	4	10.79	1
Vertical detail (LH)	1.21	65,536	1.96	16,384	2.34	4,096	3.01	1,024	3.62	256	4.67	64	6.43	16	7.59	4	13.23	1
Diagonal detail (HH)	0.42	65,536	1.03	16,384	1.32	4,096	1.59	1,024	2.05	256	2.69	64	3.75	16	4.50	4	4.55	1
Overall	2.71	196,608	4.86	49,152	5.85	12,288	7.40	3,072	9.33	768	12.45	192	16.84	48	21.29	12	58.99	4

since further compression would be more likely performed with lossy JPEG2000.

In keytagging, the choice of wavelet level, j, is very relevant. High j values permit obtaining very stable image features from the lowest frequencies. Tags retrieved from keytags associated to these features endure with little or no distortion high image compression rates and aggressive low-pass image filtering, e.g. averaging masks, Gaussian and median filters. Therefore, our choice is setting the highest possible value (Algorithms 1-2: line 4), MWL, given by the maximum number of recursive steps of decimation by 2. Furthermore, it has been observed that each time a new decomposition level is calculated, the sum of the energy of the coefficients in the four resulting subbands exceeds the energy of the coefficients in the mother subband. Thus, calculating the maximum decomposition level maximizes the number of high magnitude coefficients available for keytagging. As is shown in Table 2, the highest decomposition levels concentrate more energy, $\frac{\sum_{i,j \in subband} C(i,j)^2}{\#C(i,j) \in subband}$, from the lowest frequencies. Nonetheless, they have fewer coefficients, which results in less capacity.

To balance capacity and a suitable degree of robustness to typical image modifications in the medical context (see Section 4.1), it was determined through exhaustive testing that the optimal subset of wavelet coefficients C for the association of keytags comes from allowed wavelet levels, $aWL \geq MWL - 6$ for stable keytags (Algorithm 3: lines 17, 20), $aWL \leq MWL - 7$ for semistable (Algorithm 3: lines 23-24), and the HH subband of $WL = 1$ for volatile keytags (Algorithm 3: lines 27-28). This multiplexing of keytags in the wavelet domain is represented in Figure 1: center. In addition to this, the final aWL for stable keytags is adjusted to the length of the tag. It was empirically found that setting aWL to the maximum wavelet level that contains a number of coefficients $\geq 10 \cdot length(T)$ (Algorithm 3: lines 18-19) improves the endurance of tags to local operations such as median filtering, while maintaining the endurance to compression and common image processing. This occurs thanks to the restriction of preserving only those features coming from the lowest frequencies of the image, the most robust to low-pass filtering. Finally, the features F that will be used for keytag coding are extracted. These are the sign bit of the coefficients in C (the most robust to image changes, Algorithm 3: lines 21, 25) if the tag is stable/semistable and the LSB if the tag is volatile

(Algorithm 3: line 29).

3.3. Coding and decoding of keytags

A keytag basically encodes an input tag T, a binary string, as the positions of selected binary features F from the subset C. The extraction of C from ROI_g depends on the intended tag type (stable, semistable or volatile), as explained in Section 3.2. The coding proposed below is intended to bound the keytag with the image in a compact and fast manner. To ensure this bounding, each feature F can encode only one T bit. Otherwise, it would be known that each time that a certain feature is repeated, it encodes the same bit value, so a percentage of T bits could be derived from the keytag without the image. In addition to this, the algorithm encodes unidirectionally since changes of direction would indicate that two consecutive T bits have opposite values.

To minimize the size of the keytags, T and F are transformed into $(T)^*$ and $(F)^*$ by means of a scrambling process (Algorithm 1: lines 9-10). The intention of this scrambling is that the mean bit value of $(T)^*$ and $(F)^*$ is approximately 0.5, and that any long series of 0s or 1s is broken. To scramble the bits of T and F in a reversible manner (see Algorithm 3: line 42), they are XOR-ed with a sequence meeting Golomb's randomness postulates [44], so that the operation can be reversed by a second XOR with the same sequence. Golomb's postulates establish that in this type of random sequences 1) the number of 1s and 0s is approximately the same; 2) half of the bits in the sequence belong to a series of length 1 (e.g. ...0$\underline{1}$0..., ...1$\underline{0}$1...), a quarter of the bits in the sequence belong to a series of length 2 (e.g. ...0$\underline{11}$0..., ...1$\underline{00}$1...), an eight of the bits in the sequence belong to a series of length 3 (e.g. ...0$\underline{111}$0..., ...1$\underline{000}$1...) and so on; and 3) that the out-of-phase correlation $AC(k)$ has the same value with different values of k. The procedure to obtain a Golomb sequence consists of running a non-zero input in a linear feedback shift register [45] (see Algorithm 3: lines 42-46) with taps described by an irreducible polynomial of the finite field F_2 [46]. We chose the polynomial $x^{17} + x^3 + 1$ (see Algorithm 1: line 6) to obtain a high periodicity of $2^{17} - 1$, much larger than the size of any T to be used in the evaluation (see Section 4), and truncate the resulting Golomb sequence to the length of T/F.

As a result of scrambling, there is a probability of 0.5 of encoding a $(T_i)^*$ bit with the first available feature in $(F)^*$ (when $(T_i)^* = (F_j)^*$), a probability of $(1 - 0.5) \cdot 0.5 = 0.5^2$ of encoding it with the second available feature (if $(T_i)^* \neq (F_j)^*$, $(T_i)^* = (F_{j+1})^*$), a probability of $(1-0.5)^2 \cdot 0.5 = 0.5^3$ of encoding it with the third available feature (if $(T_i)^* \neq (F_j)^*$, $(T_i)^* \neq (F_{j+1})^*$, $(T_i)^* = (F_{j+2})^*$) and so on. According to this, the sum of the series $\sum_{k=1}^{\infty}(0.5)^k \cdot k$ gives the average number of features from $(F)^*$ required to encode a $(T)^*$ bit, 2. To guarantee high robustness, the function *buildBinaryMap* (Algorithm 1: line 11) keeps only those features of F coming from the N highest-magnitude coefficients in $abs(C)$ for the coding (see Algorithm 3: lines 31-40). N is the minimum number of features from $(F)^*$ required for the coding of any $(T)^*$ of a certain length. It has already been demonstrated that the average number of features required is $2 \cdot length(T)$, so we set $N = (2 + 6\sigma(length(T))) \cdot length(T)$, which ensures that the number of tags which can not be completely coded is < 1 for every $5 \cdot 10^8$. To set reliable values for the standard deviation, $\sigma(length(T))$, we created 10^6 random $(F)^*$ and $(T)^*$ for each length corresponding to the powers of 2 ranging from 64 to 8192 bits, and proceeded to the coding. To generate realistic random binary sequences T and F, we created individual pseudo-random float sequences $\{X\}$ with uniform distribution probability $Pr(X) \in [0, 1]$ and transformed them into binary sequences with random bias by doing $Y = ((-1)^{(X\{1\}<0.5)} \cdot 0.5 \cdot X\{2\} + X) > 0.5$. The results from the coding test showed that the mean value of N was exactly $2 \cdot (length(T))$ and that the σ for those tag lengths was $[0.1853, 0.1218, 0.0865, 0.0629, 0.0423, 0.0292, 0.0214, 0.0150]$. If a given $length(T)$ is among two of these studied values (e.g. 2048 and 4096), the σ of the lower of these two values is used. Next, $buildBinaryMap$ creates a bi-level map BM of the same size as C and initializes all its elements to be white (see Algorithm 1: line 24). This function encodes in BM the $(T)^*$ bits as features $(F_j)^*$: the first element moving forward along $(F)^*$ from the last encoding element $j - 1$ whose feature value matches the $(T_i)^*$ bit value to be encoded is marked as a black pixel in the same position in BM (see Algorithm 1: 25-38 and Figure 1: bottom). This process is repeated until all $(T)^*$ bits have been coded. Finally, for a compact arithmetic coding of BM, the standard JBIG2 [47] is applied in lossless mode (Algorithm 1: line 12). It uses a context-dependent algorithm called the QM coder. The result is the keytag KT, which will be used to reverse this process and retrieve $(T)^*$, by overlapping $(F)^*$ and BM, and obtain T (Algorithm 2: line 15).

3.4. Cryptographic protection of keytags

Since the keytagging algorithms are intended to become public, they shall include cryptographic protection for the keytags. In the first place, the tag issuer shall digitally sign his/her keytags with his/her private key for signature ($PrIssuer$, see Algorithm 1: line 13), so that any user can verify their origin and integrity with its paired public key ($PbIssuer$, see Algorithm 2: line 10). In addition to this, confidential keytags shall be encrypted with random keys ($Sk\{i\}$, see Algorithm 1: lines 14-15), which will be grouped and encrypted with the public key of each intended user ($PbUser\{i\}$, see Algorithm 1: lines 16-21). In this way, only authorised users can retrieve their symmetric keys with their private keys ($PrUser\{i\}$, see Algorithm 2: lines 7) and decrypt their authorised keytags (Algorithm 2: line 9). According to the recommendations of the NIST on long-term security (> year 2030) and the data concerning the performance of different security

Algorithm 1 Keytag association

```
procedure KEYTAGGING($I_{or}$, $\underline{f}$, $\{T\}$, $\{keytagType\}$, $PrIssuer$, $\{authorisedTags\}$, $\{PbUser\}$)
    $ROI(I_{or}) \leftarrow segmentation(I_{or}, \underline{MBR})$    ▷ To segment the $ROI$
    $ROI_g \leftarrow 0.299 \cdot R(ROI(I_{or})) + 0.587 \cdot G(ROI(I_{or})) + 0.114 \cdot B(ROI(I_{or}))$    ▷ $ROI$ to grayscale
    $MWL(ROI_g) \leftarrow \lceil log2(min(\#rows(ROI_g), \#columns(ROI_g))) \rceil$    ▷ Maximum wavelet
                                                                                        ▷ decomposition level of $ROI_g$
    $Coef, h, w \leftarrow WT(ROI_g, \underline{f}, MWL(ROI_g))$    ▷ Wavelet transformation of $ROI_g$
    $GolombSeq \leftarrow LFSR([1,0,0,0,1,1,0,1,1,1,0,1,1,0,1,1,0],[17,3])$    ▷ See Algorithm 3
    $pointers \leftarrow [1,1,1]$
    for $i$ in 1 to $\#T$ do    ▷ This loop associates one keytag in each iteration
        $(F)^* \leftarrow extractFeatures(Coef, h, w, 1, keytagType\{i\}, length(T\{i\}), MWL(ROI_g), GolombSeq)$
        $(T\{i\})^* \leftarrow scramble(T\{i\}, GolombSeq)$    ▷ See Algorithm 3
        $BM, pointers \leftarrow buildBinaryMap((F)^*, (T\{i\})^*, pointers, keytagType\{i\})$
        $KT\{i\} \leftarrow compress(BM, \underline{JBIG2})$
        $KT\{i\} \leftarrow [KT\{i\}, DS(KT\{i\}, \underline{ECDSA256}, PrIssuer)]$    ▷ Adding a signature to each keytag
        $Sk\{i\} \leftarrow createRandomKey(\underline{AES128})$
        $KT\{i\} \leftarrow encrypt(KT\{i\}, \underline{AES128}, Sk\{i\})$    ▷ Symmetric encryption of each keytag
    for $i$ in 1 to $\#PbUsers$ do    ▷ This loop prepares the cryptographic material
        $SkU \leftarrow [\ ]$    ▷ to allow users to retrieve their authorised keytags
        for $j$ in 1 to $\#T$ do
            if ($authorisedTags\{i,j\}$) then    ▷ $authorisedTags$ sets which users
                $SkU \leftarrow [SkU, Sk\{j\}]$    ▷ have access to each keytag
        $KeysUser\{i\} \leftarrow encrypt(SkU, \underline{RSA2048}, PbUser\{i\})$    ▷ Only an authorised user can decrypt
                                                                                   ▷ his/her entry with his/her $PrUser$
    return $\{KT\}$, $\{KeysUser\}$

procedure BUILDBINARYMAP($(F)^*$, $(T\{i\})^*$, $pointers$, $keytagType$)    ▷ To build the binary map
                                                                           ▷ that encodes $(T\{i\})^*$ as elements in $(F)^*$
    $BM \leftarrow zeros((F)^*)$    ▷ Initialized as a matrix of zeros with the same size as $(F)^*$
    if $keytagType = Stable$ then
        $p \leftarrow 1$
    else if $keytagType = Semistable$ then
        $p \leftarrow 2$
    else    ▷ $keytagType$ is $Volatile$
        $p \leftarrow 3$
    $index \leftarrow pointers(p)$    ▷ To continue encoding from the first available position
    for $v$ in 1 to $length((T\{i\})^*)$ do
        $r, s \leftarrow obtainIndices((F)^*, index)$    ▷ See Algorithm 3
        while $(T\{i\})^*(v) \neq (F)^*(r,s)$ do    ▷ To move forward along $(F)^*$ from the last
            $index \leftarrow index + 1$    ▷ encoding element until their values match
            $r, s \leftarrow obtainIndices((F)^*, index)$
        $BM(r,s) \leftarrow 1$    ▷ To record the position where the element $v$ in $(T\{i\})^*$
        $index \leftarrow index + 1$    ▷ matches the first available element in $(F)^*$
    $indices \leftarrow pointers$
    $indices(p) \leftarrow index$    ▷ To update the $pointer$ used
    return $BM$, $indices$
```

Algorithm 2 Tag retrieval

```
procedure tagRetrieval($I_{or}$, $\underline{f}$, $\{KT\}$, $\{keytagType\}$, $PbIssuer$, $KeysUser$, $PrUser$)
    $ROI(I_{or}) \leftarrow segmentation(I_{or}, \underline{MBR})$    ▷ To segment the $ROI$
    $ROI_g \leftarrow 0.299 \cdot R(ROI(I_{or})) + 0.587 \cdot G(ROI(I_{or})) + 0.114 \cdot B(ROI(I_{or}))$    ▷ $ROI$ to grayscale
    $MWL(ROI_g) \leftarrow \lceil log2(min(\#rows(ROI_g), \#columns(ROI_g))) \rceil$    ▷ Maximum wavelet
                                                                                      ▷ decomposition level of $ROI_g$
    $Coef, h, w \leftarrow WT(ROI_g, \underline{f}, MWL(ROI_g))$    ▷ Wavelet transformation of $ROI_g$
    $GolombSeq \leftarrow LFSR([1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0], [17, 3])$    ▷ See Algorithm 3
    $\{Sk\} \leftarrow decrypt(KeysUser, \underline{RSA2048}, PrUser)$    ▷ To obtain the symmetric keys
                                                                        ▷ of the keytags of the user
    for $i$ in 1 to $\#KT$ do    ▷ This loop retrieves a tag in each iteration
        $KT\{i\} \leftarrow decrypt(KT\{i\}, \underline{AES128}, Sk\{i\})$    ▷ To decrypt the keytag
        if $checkDS(KT\{i\}, \underline{ECDSA256}, PbIssuer)$ then    ▷ To verify its digital signature
            $KT\{i\} \leftarrow removeDS(KT\{i\})$    ▷ To remove the signature after verification
            $BM \leftarrow uncompress(KT\{i\}, \underline{JBIG2})$    ▷ To obtain the binary map that encodes a tag
            $lengthT \leftarrow sum(BM)$    ▷ The length of the tag is the number of 1s in $BM$
            $(F)^* \leftarrow extractFeatures(Coef, h, w, 0, keytagType\{i\}, lengthT, MWL(ROI_g), GolombSeq)$
    ▷ See Alg. 3
            $T\{i\} \leftarrow extractTag((F)^*, BM, GolombSeq)$
        else
            $Warning\ caused\ by\ invalid\ signature$    ▷ The tag is not retrieved if the
                                                        ▷ signature of its keytag is invalid
    return $\{T\}$

procedure extractTag($(F)^*$, $BM$, $GolombSeq$)    ▷ To extract a tag from $(F)^*$ by means of $BM$
    $(T)^* \leftarrow [\,]$
    for $r$ in 1 to $\#rows(BM)$ do
        for $s$ in 1 to $\#columns(BM)$ do
            if $BM(r, s)$ then    ▷ To move along $BM$ and find the 1$s$,
                $(T)^* \leftarrow [(T)^*, (F)^*(r, s)]$    ▷ which spot the positions in $(F)^*$
                                                           ▷ that encode the bits of $(T)^*$
    $T \leftarrow scramble((T)^*, GolombSeq)$    ▷ To retrieve the original tag, $T$
    return $T$
```

Algorithm 3 Auxiliary procedures used in keytag association and tag retrieval

```
procedure LFSR($s$,$t$)    ▷ To calculate the output of running a $LFSR$ with initial state $s$ and taps $t$
    $n \leftarrow length(s)$
    $m \leftarrow length(t)$
    $c(1, :) \leftarrow s$    ▷ $C$ stores in its rows all the states of the $LFSR$
    for $k$ in 1 to $2^n - 2$ do
        $b(1) \leftarrow s(t(1)) \oplus s(t(2))$    ▷ $b$ is used to calculate the
        if $m > 2$ then    ▷ feedback for the next state
            for $i$ in 1 to $m - 2$ do
                $b(i + 1) \leftarrow s(t(i + 2)) \oplus b(i)$
        $s(2 : n) \leftarrow s(1 : n - 1)$    ▷ Shifting the bits of the state one position
        $s(1) \leftarrow b(m - 1)$    ▷ The first element in the state is the feedback
        $c(k + 1, :) \leftarrow s$    ▷ from the previous
    $s\_outS \leftarrow c(:, n)$    ▷ The output is the concatenation of the outputs of each state
    return $s\_out$
```

Algorithm 3 Auxiliary procedures used in keytag association and tag retrieval

```
procedure EXTRACTFEATURES($Coef$, $h$, $w$, $filter$, $keytagType$, $lengthT$, $MWL$, $GolombSeq$)
    if ($keytagType = Stable$) then
        $aWL \leftarrow MWL - 6$                                          ▷ Selecting coefficients from top levels,
        while $h(aWL+1) \cdot w(aWL+1) \geq 10 \cdot lengthT$ do          ▷ such their number of coefficients
            $aWL \leftarrow aWL + 1$                                      ▷ exceeds $10 \cdot lengthT$
                                                                          ▷ The maximum number of coefficients to be
        $C \leftarrow Coef(1 : h(aWL), 1 : w(aWL))$                      ▷ selected are those from levels $\geq MWL - 6$
        $F \leftarrow (C \geq 0)$                                         ▷ and the features of interest are their signs
    else if ($keytagType = Semistable$) then
        $aWL \leftarrow MWL - 7$                                          ▷ Selecting coefficients from levels $\leq MWL - 7$
        $C(1 : h(aWL+1), 1 : w(aWL+1)) \leftarrow 0$                     ▷ by setting coefficients from levels $\geq MWL - 6$ to 0,
        $F \leftarrow (C \geq 0)$                                         ▷ the features of interest are their signs
    else                                                                  ▷ $keytagType$ is $Volatile$
        $aWL \leftarrow 1$                                                ▷ Selecting coefficients from
        $C \leftarrow Coef(h(aWL+1)+1 : end, w(aWL+1)+1 : end)$          ▷ the $HH$ subband of level 1,
        $F \leftarrow LSB(C)$                                             ▷ the features of interest are their $LSB$s
    $(F)^* \leftarrow scramble(F, GolombSeq)$
    if $filter$ then      ▷ Removing the features from the $2 + 6 \cdot \sigma(lengthT)$ lowest magnitude coefficients
        $\sigma s = [0.1853, 0.1218, 0.0865, 0.0629, 0.0423, 0.0292, 0.0214, 0.0150]$
        $lengths = [64, 128, 256, 512, 1024, 2048, 4096, 8192]$
        for $i$ in 1 to $length(\sigma s)$ do                            ▷ This loop sets the value of $\sigma$
            if $lengthT \leq lengths(i)$ then                             ▷ according to the value of $lengthT$
                $\sigma \leftarrow \sigma s(i)$
        $sorted\_values, sorted\_indices \leftarrow sort(abs(C(:)), descending)$      ▷ Obtaining the indices of
        $indicesToDelete \leftarrow sorted\_indices(\lceil (2 + 6 \cdot \sigma) \cdot lengthT \rceil : end)$      ▷ the lowest magnitude
        $rows\_IndicesToDelete, cols\_IndicesToDelete \leftarrow obtainIndices(indicesToDelete, C)$      ▷ coefs,
        $(F)^*(rows\_IndicesToDelete, cols\_IndicesToDelete) \leftarrow 2$      ▷ which are removed by setting them
    return $(F)^*$                                    ▷ to 2, a value not existing in $T$ (composed of 0s and 1s)

procedure SCRAMBLE($M$,$GolombSeq$)                                       ▷ To scramble/descramble a binary vector
    $(M)^* \leftarrow M$                                                  ▷ or matrix $M$ by adding $GolombSeq$
    for $i$ in 1 to $\#rows(M)$ do
        $(M)^*(i, :) \leftarrow M(i, :) \oplus GolombSeq(1 + (i-1) \cdot \#columns(M) : i \cdot \#columns(M))$
    return $(M)^*$

procedure OBTAININDICES($M$, $pointers$)                                  ▷ To translate unidimensional $pointers$ into
    $r \leftarrow \lceil pointers/\#columns(M) \rceil$                    ▷ bidimensional indices for a matrix $M$
    $s \leftarrow 1 + module(pointers - 1, \#columns(M))$
    return $r$, $s$                                   ▷ $r$ are de indices for rows, $s$ are the indices for columns
procedure ZEROS($M$)                                                      ▷ To initialize a vector/matrix of zeros
    $zerosM \leftarrow M$                                                 ▷ of the same size as $M$
    $zerosM(1 : end, 1 : end) \leftarrow 0$
    return $zerosM$
procedure SUM($M$)                                                        ▷ To sum all the elements in a matrix/vector $M$
    $S \leftarrow 0$
    for $i$ in 1 to $\#rows(M)$ do
        for $j$ in 1 to $\#columns(M)$ do
            $S \leftarrow S + M(i, j)$
    return $S$
```

algorithms provided by [48], these are the choices for the following:

- Digital signature of a keytag KT: The standardized Elliptic Curve Digital Signature Algorithm 256 [49], which performs signature-verification slightly faster ($3.92 - 6.56\,Mcycles$) than the other two algorithms authorised by the NIST, DSA [50] and RSA [51] with a 2048-bit key length. DSA was replaced by ECDSA because the latter operates with smaller numbers, and thus it is hard nowadays to find implementations of DSA supporting 2048 bits. On the other hand, RSA2048 has a similar overall performance ($11.06 - 0.29\,Mcycles$), but ECDSA produces a much shorter signature (512 bits vs 2048) with a roughly similar security level. Digital signature keys, $PbIssuer$-$PrIssuer$, shall be renewed every 1-3 years. As part of the checking of a digital signature, it is required to verify that the digital certificate of the signatory has not expired or been revoked, by means of Check Revocation Lists or by using the Online Certificate Status Protocol [52].
- Symmetric encryption of a keytag KT: the Advanced Encryption Standard (AES [53]), chosen by the NIST after a contest [54] with five finalists: MARS, RC6, Rjindael, Serpent and Twofish. Rjindael, designed by Joan Daemen and Vicent Rijmen was the winner, and it became the AES after some adjustments. It is considered the most secure symmetric algorithm and it is also the fastest of the five AES finalists, with an encryption/decryption speed of 12.6 cycles/byte. Regarding overhead, the key size is set to 128 bits and its block size is also 128 bits, so padding bits will be added if the data to be encrypted is not a multiple of this block size. A symmetric key shall be created to encrypt each keytag.
- Asymmetric encryption of authorised users' access keys SkU: RSA2048 [51], Public Key Cryptographic Standard #1 published by RSA Security Inc. The algorithm elGamal [55] would be the alternative, but RSA2048 is preferred because elGamal is not a standard. The block size of RSA2048 is 2048 bits, so there will be overhead if the data to be encrypted is not a multiple of this block size, and its performance for encryption-decryption is $0.29 - 11.22\,Mcycles$ per block. Asymmetric encryption keys, $PbUsers$-$PrUsers$, shall be renewed every 1-2 years.

3.5. Support for JPEG2000 compression

The JPEG2000 algorithm is a standard for the efficient representation and interchange of digital images with different characteristics (scientific, medical, rendered graphics, etc.), allowing different imaging models, e.g. client/server, real-time transmission, image library archival, limited buffer and bandwidth resources. It was created by members of the Joint Picture Experts Group, comprising members of the International

Telecommunication Union Standardization Sector and the International Standardization for Organization. JPEG2000 provides low bit-rate operation with rate-distortion and improves the subjective image quality performance of the previous standard, JPEG. Although this standard is still not as widely used for natural images as its forerunner, it is widespread in medical imaging and included in the DICOM standard. According to the mean opinion score (MOS) of medical experts [56], this codec maintains good clinical quality at compression ratios up to 8-16 for magnetic resonance, ultrasound and X-ray images. Compared to JPEG, it presents a better image distortion-rate tradeoff and for an equal objective distortion (e.g. $PSNR = 35\,dB$), it obtains a higher MOS.

Both this compressor and the keytagging process share the use of wavelets, with the same filters, for a representation of the image adjusted to the properties of the human vision system. As a result, JPEG2000 compression causes no or very little distortion to the retrieved tags, as is demonstrated in Section 4.4. In addition to this, if the keytags association is integrated in the compression process, most of the runtime cost of the former is covered by the latter (see Section 4.5). Finally, JPEG2000 formats .jp2 and .jpx allow the embedding of metadata, which can be used to conveniently store the keytags.

To ensure full compliance of compression with keytagging, the initial color transformation must be the irreversible TIC and the further tiling of the image must be set to its size. The last steps of the compression are independent from the keytagging. These are performing context modeling and bit-plane arithmetic coding, arranging the coded data in layers corresponding to quality levels and performing post-compression rate allocation.

4. Experimental evaluation

The features of this algorithm that need to be experimentally evaluated are its robustness-capacity tradeoff when the image undergoes different image modifications, its specificity when the original image is replaced, its compatibility with JPEG2000 compression, its average runtime cost for different parameter configurations and its scalability when the image size is increased. Complementarily, Section 5 analyzes the security foundations of the keytagging method.

4.1. Evaluation setup

The image test set is composed of 64 images, sized $512 \times 512\,px^2$, corresponding to different medical modalities and parts of the body:

- 18 computed tomography (CT) images, gathered from [57]: 6 chest (Artifix), 6 dental area (Incisix) and 6 pelvis (Pelvix) images.
- 18 magnetic resonance images (MRI), gathered from [57]: 6 brain (Brainix), 6 knee (Knix) and 6 thoracic and lumbar area (MRIX) images.
- 12 positron emission tomography (PET)-CT images from a whole body scan, gathered from [57] (PETCETIX).
- 16 ultrasound images (US): 4 mode B echocardiograms provided by Lozano Blesa Hospital in Zaragoza, 4 mode M, 4 Doppler color, 4 pulsed and continuous wave Doppler.

The image test set is processed with typical modifications in the medical context, which are indexed for reference in Figure 2 and Tables 3 and 5:

- 1-10. Compression: JPEG with quality factors 75%, 50%, 25%, 15% and 5% (Figure 2: 5), JPEG2000 with compression ratios 4:1, 8:1, 16:1, 32:1 and 64:1 (Figure 2: 10).
- 11-18. Common image processing: β correction -0.3, -0.5 (Figure 2: 12), $+0.4$ and $+0.7$ (Figure 2: 14), contrast stretching 2% and 10% (Figure 2: 16), color inversion (Figure 2: 17) and local histogram equalization (Figure 2: 18).
- 19-27. Local operations: edge sharpening (Figure 2: 19), median filtering 5×5 and 7×7 (Figure 2: 21), averaging mask 5×5 and 7×7 (Figure 2: 23), Gaussian filtering 7×7 and 11×11, and motion blur with 7 and 9 pixels displacement (Figure 2: 27).
- 28-33. Geometric transformations: clipping the ROI, rotating 90, 180 and 270, horizontal and vertical flipping (Figure 2: 33).
- 34. Insertion of visible annotations (Figure 2: 34).
- 35. Blackening private data parts for anonymization.

and attempting to distort the tag or part of it, by means of a watermark-based attack:

- 36-40. Modification of $l = 64$ (Figure 2: 34), $l = 128$, $l = 256$, $l = 512$, $l = 1024$ and $l = 2048$ sign bits from the highest-level wavelet coefficients of the image.

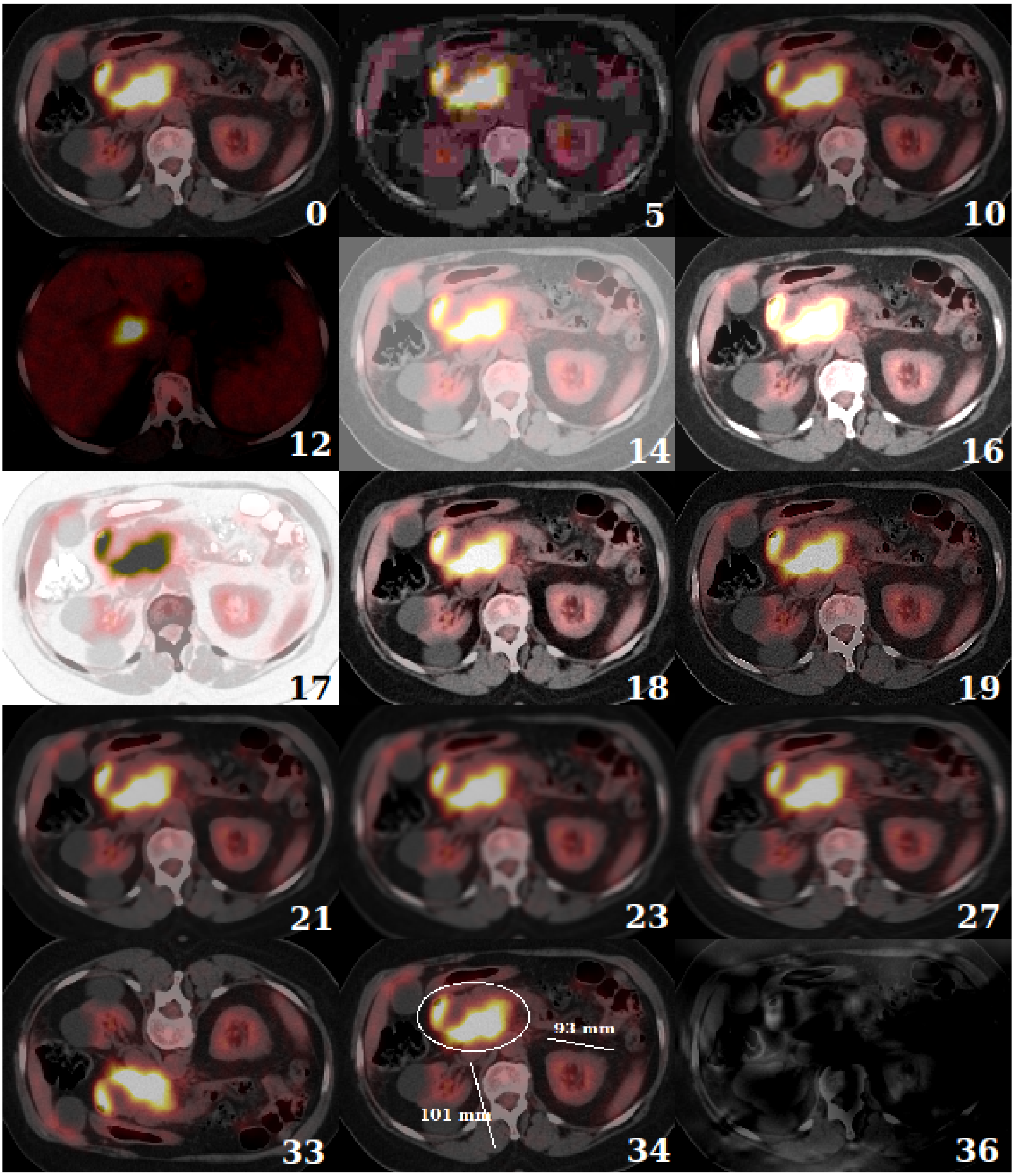

Figure 2: *ROI* of a $512 \times 512\,px^2$ PET-CT image, original (0) and resulting from the application of JPEG QF=5% and JPEG2000 CR 64:1 compression (5,10), β correction -0.5 and $+0.7$ (12,14), 10% contrast stretching (16), color inversion (17), local histogram equalization (18), edge sharpening (19), median filter 7×7 (21), image averaging 7×7 (23), motion blur 9 (27), vertical flipping (33), insertion of annotations (34) and watermark-based attack with l=128 (36).

The robustness of keytagging to those modifications is evaluated by measuring the distortion of the retrieved tags $\tilde{T}$ with respect to the original T associated to I_{or}, by means of the Normalized Hamming Distance [58]:

$$NHD = \frac{\tilde{T} \oplus T}{length(T)}, \tag{1}$$

where T and $\tilde{T}$ are binary vectors and $\oplus$ is the XOR logical operator. Those $\tilde{T}$ that endure the image modifications end up very similar to T and obtain $NHD \approx 0$, while those $\tilde{T}$ which are very dissimilar to T obtain $NHD \approx 0.5$, e.g. when retrieved from very degraded versions of I_{or}. The Normalized Hamming Distance is very useful to determine how much redundancy needs to be added to overcome the distortion of $\tilde{T}$ by means of some redundant coding.

Simulation setup 1: The image modifications described above are applied to the test set. A fixed-length random tag T is associated to each image, by means of keytags KT, retrieving $\tilde{T}$ from the corresponding modified image $\tilde{I_{or}}$. The resulting NHD is calculated, and the process of keytag association-tag retrieval is repeated for different tag lengths and for the three types of keytags: stable, semistable and volatile. The results are depicted in Table 3, which also shows how much distortion is caused to ROI_g (the area of I_{or} used for the association of keytags) by each modification (which results in a $\tilde{ROI_g}$), measured with two different indices, the classic Peak Signal-to-Noise Ratio:

$$PSNR(I, \tilde{I}) = 20 \cdot log_{10}\left(\frac{max(I(:))}{\sqrt{\frac{1}{\#rows(I) \cdot \#columns(I)} \cdot \sum_{i=1}^{\#rows(I)} \sum_{j=1}^{\#columns(I)} ||I(i,j) - \tilde{I}(i,j)||^2}}\right), \tag{2}$$

and also with the mean Structural SIMilarity index:

$$MSSIM(I, \tilde{I}) = \frac{1}{M} \sum_{j=1}^{M} \frac{(2 \cdot \mu_{I_j}\mu_{\tilde{I}_j} + (0.01 \cdot L)^2) \cdot (2 \cdot \sum_{i=1}^{N} w_i \cdot (I_i - \mu_I)(\tilde{I}_i - \mu_{\tilde{I}}) + (0.03 \cdot L)^2)}{(\mu_{I_j}^2 + \mu_{\tilde{I}_j}^2 + (0.01 \cdot L)^2) \cdot (\sum_{i=1}^{N} w_i \cdot ((I_i - \mu_I)^2 + (\tilde{I}_i - \mu_{\tilde{I}})^2) + (0.03 \cdot L)^2)}, \tag{3}$$

where L is the dynamic range of the pixel values (255 for 8-bit grayscale images), w_i correspond to an 11×11 circular-symmetric Gaussian weighting function with standard deviation 1.5 samples (normalized so that $\sum_{i=1}^{N} w_i = 1$), $\mu_I = \sum_{i=1}^{N} w_i \cdot I_i$, M is the number of local windows in the image and N the number of pixels in the local window. Further details about this index may be consulted in [59] and the implementation of the $MSSIM$ algorithm used in this work is available online at [60].

The $PSNR$ is a simple mathematical measure that directly compares the value of the pixels from the two images. Although it is very popular, the correlation between this measure and the visual perception of quality is not tight enough in many cases. The $MSSIM$ assumes that the human vision system is highly adapted for extracting structural information from images. Thus, it basically compares local patterns of pixel intensities that have been normalized for luminance and contrast.

Simulation setup 2: This follows the process of setup 1 but considers each image as a modified version of the rest. This setup is intended to evaluate the degree of distortion of tags retrieved from images that are different from the original ones. Since some images of the test set come from the same patient and acquisition session, some pairs of ROI_g are quite similar: five have $PSNR > 20\,dB$ and $MSSIM > 0.67$, and may obtain low NHD values.

4.2. Robustness-capacity

In the case of stable and semistable tags, there is a tradeoff between their lengths and their robustness to image modifications. Naturally, this occurs because the keytag association algorithm sorts and selects image features according to their degree of robustness, in descending order. Nonetheless, not all the modifications have the same impact on the image. Figure 1 illustrates the effect of several modifications from Section 4.1 on a medical image, and Table 3: column 2 (unshaded cells) shows the average $PSNR$ and $MSSIM$ of the image test set after each modification. JPEG2000 compression is one of the most typical modifications. It maintains the image clinical value with compression factors around 16:1, obtaining $PSNR \simeq 42\,dB$ and $MSSIM \simeq 0.99$. Higher compression ratios may cause unacceptable distortion, so they are not recommended. Common image processing techniques can help to find a better representation of the image. β correction changes the brightness and can be reversed; contrast stretching, local histogram equalization and color inversion help to enhance the details of the image, and the latter can be totally reversed. In the case of local operations, edge sharpening helps to enhance certain details, so it only modifies volatile parts of the image. The rest of the local operations modify both the semistable and volatile parts of the image, high and some middle frequencies, leaving only the stable parts intact. There are also several image modifications that neither affect the image quality nor distort its associated tags. These are clipping the ROI, since there are no keytags associated outside this region; geometrical changes, which are detected and reversed by means of a resynchronization step depicted in Section 5.2; inserting annotations, usually in the borders of the ROI or outside; and darkening private data, which is most often located outside the ROI. Finally, it was observed that modifying the sign of the most significant coefficients in the highest decomposition level is the most effective manner to willfully destroy stable tags, but at the cost of completely destroying the image as well, since the $PSNR$ becomes ≤ 13.

Table 3: Average distortion of variable-length stable, semistable and volatile tags when the image test set (unshaded cells) and its interpolated counterpart (lightly shaded cells when calculating aWL according to Algorithm 1, shaded cells when maintaining the aWL of the original test set) undergo common modifications in the medical context (see Sections 4.1 and 4.6).

	Image size —px^2—	Image quality —$PSNR(dB)$, $MSSIM$—	Distortion of stable tags —NHD (%)—					Distortion of semistable tags —NHD (%)—								Distortion of volatile tags —NHD (%)—		
#. Operation / $Length(T)$	-	-	128/BCH	256/BCH	512/BCH	1024/BCH	2048	128/BCH	256/BCH	512/BCH	1024/BCH	2048/BCH	4096/BCH	8192/BCH	16384/BCH	64	128	256
Compression																		
1. JPEG QF=75%	512 × 512	37.7, 0.98	0	0	0	0	0	0	0	0	0	0.2/0	1.2/0.3	4.2/10.6	11.8/19.8	48.4	48.4	49.4
JPEG QF=30%	1024 × 1024	38.9, 0.98	0	0	0	0	0	0	0	0	0	0	0.4/0	2.4/4.5	6.5/13.8	50.8	50	49.8
JPEG QF=30%	1024 × 1024	38.9, 0.98	0	0	0	0	0	0	0	0	0.3/0	1/0	2.7/7.9	8.2/14.9	15.9/23.5	50.8	50	49.8
2. JPEG QF=50%	512 × 512	35.2, 0.97	0	0	0	0	0	0	0	0	0.3/0	1.2/0	3.9/7.7	8.7/16.6	18.5/26.1	48.4	49.2	49.8
JPEG QF=20%	1024 × 1024	37.0, 0.97	0	0	0	0	0	0	0	0	0	0.2/0	1.2/0.4	4.5/10.9	11.1/18.8	49.2	50.2	50.1
JPEG QF=20%	1024 × 1024	37.0, 0.97	0	0	0	0	0	0	0	0.2/0	1/0	2.5/4.4	6.3/13.1	13.6/22	21.3/29.6	49.2	50.2	50.1
3. JPEG QF=25%	512 × 512	33.2, 0.95	0	0	0	0	0.2	0	0	0.6/0	1.4/0	4/7.8	8.8/17.2	17.2/26.3	26.3/33.5	50	49.6	50.8
JPEG QF=15%	1024 × 1024	35.3, 0.92	0	0	0	0	0.2	0	0	0	0.1/0	0.6/0	2.5/7.9	8.5/14.9	14.9/22.3	50	49.2	49.9
JPEG QF=15%	1024 × 1024	35.3, 0.92	0	0	0	0	0	0	0	0.4/0	1.4/0	4.2/9.1	10.1/17.2	18.5/26.2	25.8/32.6	50	49.2	49.9
4. JPEG QF=15%	512 × 512	31.4, 0.92	0	0	0	0	1.9	0	0.8/0	1.4/0	4.7/7.4	9.7/15.3	17/25	26.6/33.8	34/38.6	49.2	50.8	49.6
JPEG QF=10%	1024 × 1024	33.2, 0.92	0	0	0	0	1	0	0	0	0.5/0	2.2/4.4	6.7/13.7	14.2/23.4	23.9/31.5	50.8	49.8	49.8
JPEG QF=10%	1024 × 1024	33.2, 0.92	0	0	0	0	0.3	0	0.4/0	1.4/0	3.9/8.2	8.7/15.2	15.9/23.7	23.8/31.3	32.2/37.5	50.8	49.8	49.8
5. JPEG QF=5%	512 × 512	26.7, 0.80	0	0	0.7/0	4.5/5.9	13.6	3.5/10.8	7.2/10.2	12.6/16.6	18.8/23.9	27/32.6	34.3/39.2	40.1/43.9	44.5/45.8	46.9	50	49.8
JPEG QF=5%	1024 × 1024	28.9, 0.74	0	0	0	0/2.6	6.3	0	0.4/0	1.4/0	4.3/10.4	10.5/17.4	18.6/26.4	26.9/34	35/39.4	50.8	50.8	49.7
JPEG QF=5%	1024 × 1024	28.9, 0.74	0	0	0	0.9/0	3.8	3.1/9.1	4.5/10	8/15.6	14/20.9	20.6/27.6	27.6/33.3	34.1/39.3	40.2/43.5	50.8	50.8	49.7
6. JPEG2000 CR 4:1	512 × 512	50.5, 1.0	0	0	0	0	0	0	0	0	0	0	0	0	0	51.6	50	50
JPEG2000 CR 16:1	1024 × 1024	50.2, 1.0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	50	50
JPEG2000 CR 16:1	1024 × 1024	50.2, 1.0	0	0	0	0	0	0	0	0	0	0	0	0	0.3/0	50	50	50
7. JPEG2000 CR 8:1	512 × 512	47.6, 0.99	0	0	0	0	0	0	0	0	0	0	0	0	1.7/7.9	50	49.2	50.2
JPEG2000 CR 32:1	1024 × 1024	47.3, 1.0	0	0	0	0	0	0	0	0	0	0	0	0	0.7/0.4	50	49.2	50.2
JPEG2000 CR 32:1	1024 × 1024	47.3, 1.0	0	0	0	0	0	0	0	0	0	0	0	0.3/0	2.7/9.7	50	49.2	50.2
8. JPEG2000 CR 16:1	512 × 512	42.7, 0.99	0	0	0	0	0	0	0	0	0	0	0.2/0	1.7/7.2	8.7/16.1	51.6	49.2	50.4
JPEG2000 CR 64:1	1024 × 1024	42.0, 0.99	0	0	0	0	0	0	0	0	0	0	0.1/0	1.6/4.3	6.4/13.7	50	49	50.4
JPEG2000 CR 64:1	1024 × 1024	42.0, 0.99	0	0	0	0	0	0	0	0	0	0.2/0	1.6/1.3	5.2/12	12.2/20	50	49	50.4
9. JPEG2000 CR 32:1	512 × 512	37.9, 0.96	0	0	0	0	0.2	0	0	0	0	0.9/1.2	5.1/11.9	12.3/21.8	21.3/29.7	49.2	48.4	48.8
JPEG2000 CR 128:1	1024 × 1024	37.8, 0.96	0	0	0	0	0.2	0	0	0	0.1/0	0.9/0.4	4.6/11.9	11.4/21.4	21.3/29.8	50	49.8	50.2
JPEG2000 CR 128:1	1024 × 1024	37.8, 0.96	0	0	0	0	0	0	0	0.7/0	2.5/8.1	7/14.7	12.5/22.4	20.1/30.4	28.1/36.4	50	49.8	50.2
10. JPEG2000 CR 64:1	512 × 512	33.6, 0.92	0	0	0	0.2/0	3.3	0	0	0.4/0	2.8/7	7.8/16.5	16.4/25.9	27.5/34.1	35.2/39.6	51.6	50	50
JPEG2000 CR 256:1	1024 × 1024	33.7, 0.91	0	0	0	0.1/0	3.4	0	0	0.3/0	2.4/5.8	7.8/15	16.5/25.3	26.4/33.7	34.4/39.2	50	49.6	49.7
JPEG2000 CR 256:1	1024 × 1024	33.7, 0.91	0	0	0	0.2/0	2.3	2.3/7	4.5/7.2	7.3/11	13.1/18.5	19.7/27.6	27/35.5	33.6/40.4	39.8/43.8	50	49.6	49.7
Common image processing																		
11. β correction −0.3	512 × 512	20.7, 0.85	0	0	0	0	0.3	0	0	0	0	0	0	0.2/0	0.7/0	50	49.6	49.8
β correction −0.3	1024 × 1024	21.1, 0.80	0	0	0	0	0.3	0	0	0	0	0	0	0.2/0	0.6/0	48.8	49.6	48.6
β correction −0.3	1024 × 1024	21.1, 0.80	0	0	0	0	0	0	0	0	0	0	0.2/0	0.4/0	0.8/0	48.8	49.6	48.6
12. β correction −0.5	512 × 512	16.1, 0.60	0	0.4/0	0.2/0	0.8/0	2	0	0	0.1/0	0.2/0	0.5/0	1.1/0	2.1/0.7	4.6/8.8	50	50	49.6
β correction −0.5	1024 × 1024	16.3, 0.50	0	0.2/0	0.2/0	0.7/0	2.1	0	0	0.1/0	0.3/0	0.7/0	1.3/0	2.9/0.6	5/5.3	50	50.8	49.8
β correction −0.5	1024 × 1024	16.3, 0.50	0	0.4/0	0.2/0	0.7/0	1	0	0.8	1.1/0	1.3/0	1.8/0	3/0.3	4.3/5.7	7.1/10.4	50	50.8	49.8
13. β correction +0.4	512 × 512	17.1, 0.74	0	0	0	0	0.1	0	0	0	0	0.1/0	0.2/0	0.2/0	0.3/0	50	49.2	49
β correction +0.4	1024 × 1024	17.1, 0.71	0	0	0	0	0.1	0	0	0	0	0	0.2/0	0.3/0	0.4/0	47.7	48	48.7
β correction +0.4	1024 × 1024	17.1, 0.71	0	0	0	0	0	0	0	0	0	0.2/0	0.3/0	0.4/0	0.5/0	47.7	48	48.7
14. β correction +0.7	512 × 512	10.7, 0.49	0	0	0.1/0	0.5/0	1	0	0	0.2/0	0.6/0	0.8/0	1/0	1.4/0	1.6/0	51.6	49.6	50.4
β correction +0.7	1024 × 1024	10.7, 0.45	0	0	0.2/0	0.5/0	1.1	0	0	0.2/0	0.5/0	1/0	1.2/0	1.7/0	2.2/0	49.2	50.4	50.1
β correction +0.7	1024 × 1024	10.7, 0.45	0	0	0.2/0	0.5/0	0.5	0	0.8/0	1/0	1.4/0	1.7/0	1.9/0	2.3/0	2.8/0	49.2	50.4	50.1
15. Contrast stretching 2%	512 × 512	18.1, 0.87	0	0	0	0.1/0	0.2	0	0	0	0.2/0	0.3/0	0.6/0	0.9/0	1/0	50	49.2	49.2
Contrast stretching 2%	1024 × 1024	18.1, 0.72	0	0	0	0.1/0	0.2	0	0	0.2/0	0.2/0	0.7/0	1/0	1.3/0	1.5/0	49.6	49.4	49
Contrast stretching 2%	1024 × 1024	18.1, 0.72	0	0	0	0.1/0	0.2	1.6/0	0.4/0	0.5/0	0.8/0	1.7/0	1.9/0	1.9/0	1.9/0	49.6	49.4	49
16. Contrast stretching 10%	512 × 512	16.6, 0.83	0	0	0	0.3/0	0.5	0	0	0.1/0	0.3/0	1.5/0	1.7/0	1.8/0	1.9/0	47.7	50	50
Contrast stretching 10%	1024 × 1024	16.2, 0.66	0	0	0.2/0	0.2/0	0.7	0	0	0.3/0	1.1/0	1.5/0	2.2/0	2.7/0	3/0	50	50	49.8
Contrast stretching 10%	1024 × 1024	16.2, 0.66	0	0	0.2/0	0.5/0	0.5	1.6/0	1.4/0	1.5/0	3.5/0	2.9/0	3.4/0	3.5/0	3.7/0	50	50	49.8
17. Invert colors	Any size	2.7, -0.10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18. Local hist. equal.	512 × 512	21.3, 0.86	0	0	0	0.1/0	0.3	0	0	0	0	0.1/0	0.3/0	0.5/0	0.7/0	50	50	50
Local hist. equal.	1024 × 1024	21,2, 0.81	0	0	0	0.1/0	0.4	0	0	0	0.1/0	0.1/0	0.4/0	0.8/0	1.1/0	49.6	50.2	49.4
Local hist. equal.	1024 × 1024	21,2, 0.81	0	0	0	0.1/0	0.1	0	0	0.2/0	0.2/0	0.6/0	1/0	1.5/0	1.7/0	49.6	50.2	49.4

Table 3: Average distortion of variable-length stable, semistable and volatile tags when the image test set (unshaded cells) and its interpolated counterpart (lightly shaded cells when calculating aWL according to Algorithm 1, shaded cells when maintaining the aWL of the original test set) undergo common modifications in the medical context (see Section 4.1 and 4.6).

	Image size —px^2—	Image quality —$PSNR(dB)$, $MSSIM$—	Distortion of stable tags —NHD (%)—					Distortion of semistable tags —NHD (%)—								Distortion of volatile tags —NHD (%)—		
$Length(T)$ #. Operation	-	-	128/BCH	256/BCH	512/BCH	1024/BCH	2048	128/BCH	256/BCH	512/BCH	1024/BCH	2048/BCH	4096/BCH	8192/BCH	16384/BCH	64	128	256
Local operations																		
19. Edge sharpening	512 × 512	24.4, 0.89	0	0	0	0.1/0	0.5	0	0	0	0	0.2/0	0.4/0	0.6/0	1.2/0	48.4	48.8	48.8
Edge sharpening	1024 × 1024	25.4, 0.89	0	0	0	0.1/0	0.5	0	0.6/0	0.5/0	0.8/0	0.8/0	1.1/0	1.7/0	2.4/0.1	49.2	49.4	49.7
Edge sharpening	1024 × 1024	25.4, 0.89	0	0	0	0.1/0	0.3	1.6/0	2.3/0	2.8/0	2.6/0	2.7/0	2.8/0	3.5/0.6	4.6/2.7	49.2	49.4	49.7
20. Median filter 5 × 5	512 × 512	27.8, 0.92	0	0	0	0.2/0	1	5.5	5.7/6.8	9.1/11	13.1/16.3	20.4/22	27.3/28.5	31.8/33.4	35.8/37.7	50	49.2	50
Median filter 11 × 11	1024 × 1024	27.8, 0.92	0	0	0	0.4/0	1.6	4.3/6.6	5.9/7.8	10.6/13.6	15.6/17.7	23.3/25.3	29.9/31.6	35.7/36.3	39.1/39.9	48.4	50.4	49.8
Median filter 11 × 11	1024 × 1024	27.8, 0.92	0	0	0	0.4/1	5.7	41/43.8	40.4/42.7	40.5/43.9	41.2/44.8	42/47.3	44.7/48.5	46.6/49.2	47.2/47.9	48.4	50.4	49.8
21. Median filter 7 × 7	512 × 512	25.5, 0.87	0	0	0.8/0	1.9/0	4.8	23.4/23.2	25/24.5	21.6/28.3	27/32.5	32/37.2	36.3/40.5	39.4/42.4	41.5/44.3	50.8	50	50
Median filter 14 × 14	1024 × 1024	25.4, 0.88	0	0	1/0	2.1/0	4.8	22.3/22.8	21.3/23.8	23.1/27.8	25.7/32.2	32.6/36.6	37.2/40.4	40.3/43.6	43.7/45.2	50	50	49.8
Median filter 14 × 14	1024 × 1024	25.4, 0.88	0	0	1/0	2.3/9.1	13.4	46.1/45.2	46.1/48.2	45.9/47.6	45.6/47	47.2/47.9	47.1/48	47.8/47.4	47.7/49.2	50	50	49.8
22. Image averaging 5 × 5	512 × 512	26.2, 0.91	0	0	0	0.2/0	1	16.8/17.4	29.1/22.3	32.8/28.1	36.7/36.6	40.7/37.9	45.5/44.4	49.2/45.9	50/46.9	49.2	49.6	50.4
Image averaging 11 × 11	1024 × 1024	26.3, 0.90	0	0	0.2/0	0.5/0	1.8	12.1/18.9	18.6/19.7	25.1/27.2	32/35.7	40.6/41.5	47.9/46.9	51.7/49.3	51.7/49.9	50.8	50.4	49.6
Image averaging 11 × 11	1024 × 1024	26.3, 0.90	0	0	0.1/0	0.6/6.1	11.5	77.3/77.5	74.8/77.9	72/73.9	69/70.1	64.7/65	63/62.9	61.2/58.8	56.6/57.4	50.8	50.4	49.6
23. Image averaging 7 × 7	512 × 512	23.8, 0.85	0	0	1.3/0	2.6/0.3	5.5	53.9/49.6	49.4/51.9	50.9/53.5	50.3/54.9	54.7/55.7	54.8/55.1	54.1/54.4	54/53.1	48.4	48.8	50.2
Image averaging 13 × 13	1024 × 1024	25.1, 0.87	0	0	0.6/0	1.7/0	4.1	35.9/35.7	35.4/36.5	39.3/41.7	44.6/49.5	50.5/54.1	52/54.9	55.5/55.2	54.4/53.4	50.4	50.6	50.6
Image averaging 13 × 13	1024 × 1024	25.1, 0.87	0	0	0.8/0	1.9/12.5	19.7	81.3/76.4	80.5/77.1	77.6/74.4	73.2/70.2	67.3/67.7	63.1/65.7	60.9/61.2	58.1/54.9	50.4	50.6	50.6
24. Gaussian filtering 7 × 7	512 × 512	25.7, 0.90	0	0	0.2/0	0.5/0	1.7	3.1/3.3	4.5/5.5	7.3/10.9	12.4/17.6	20.5/22	25.5/29.3	31.4/33.4	36.3/38.1	50	50	49.6
Gaussian filtering 14 × 14	1024 × 1024	26.2, 0.90	0	0	0	0.3/0	1.3	0	0.8/0	2.3/1	4.9/6.1	8.9/10.2	12.2/16.8	17/21.1	23.8/26.4	50.8	50.4	50.6
Gaussian filtering 14 × 14	1024 × 1024	26.2, 0.90	0	0	0	0.4/0	2.9	3.5/4.3	3.9/6.9	7.6/11.1	12.4/15.8	16.8/22.1	20.8/26.1	26.6/32	32.5/35.5	50.8	50.4	50.6
25. Gaussian filtering 11 × 11	512 × 512	24.4, 0.89	0	0	0.4/0	1.1/0	2.9	4.7/6.1	6.3/6.8	9.6/12	14.3/20.7	23.8/26	29.2/30.7	33.4/35.5	38.1/39.2	51.6	50.4	50.2
Gaussian filtering 17 × 17	1024 × 1024	25.1, 0.87	0	0	0	0.1/0	0.9	0.8/0	2/0	4/3.6	6.7/6.9	10.4/11.7	14.3/17.7	19.3/22.8	25/27.7	50.8	50	49.4
Gaussian filtering 17 × 17	1024 × 1024	25.1, 0.87	0	0	0	0.2/0	2.9	5.5/7.9	7.8/9.5	10.2/13.5	14.3/17.9	19.2/24.7	23.5/28.3	27.6/33.2	32.9/36.8	50.8	50	49.4
26. Motion blur 7	512 × 512	27.7, 0.92	0	0	0.4/0	0.8/0	1.7	24.6/19.5	20.3/20.3	21.4/21.1	22.2/22.9	23.7/24.3	27.6/29	30.2/31.2	32.9/32.8	51.6	50.8	50
Motion blur 15	1024 × 1024	27.3, 0.92	0	0	0.4/0	1.2/0	2.7	25.8/25.2	23/24.6	23.1/22.1	23.1/22.1	23/24.3	24.5/26.2	27.3/29	30.1/30.3	49.2	49.6	49.4
Motion blur 15	1024 × 1024	27.3, 0.92	0	0	0.6/0	1.5/0.6	11.4	25.8/24.6	24.8/21.8	21.3/20.8	22/23.2	24.2/24.6	25.5/26.9	28.8/28.3	28.9/29	49.2	49.6	49.4
27. Motion blur 9	512 × 512	26.0, 0.88	0	0	2.1/0	3.6/0.2	6.3	33.2/32.2	31.6/30.5	27.1/28.4	29.5/28.6	29.6/29.9	32.5/33.6	35/36.1	36.3/35.7	51.6	50.8	50.4
Motion blur 17	1024 × 1024	26.7, 0.89	0	0	1.2/0	2.5/0	4.8	30.9/28.3	27.9/27	27.1/26.5	26.1/25.9	25.1/26.6	26.5/29	29.7/30.2	31.8/32.5	51.2	50	50.3
Motion blur 17	1024 × 1024	26.7, 0.89	0	0	1.4/0	2.9/1.5	15.7	18.8/18	17.8/18.9	20.3/20.4	21.2/22.6	23.1/25.7	25.4/27.7	28.1/29.7	30.5/31.1	51.2	50	50.3
Geometrical transformations																		
28. Clipping the *ROI*	Any size	Inf, 1.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29. Rotating 90°	Any size	12.2, 0.26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30. Rotating 180°	Any size	12.6, 0.29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31. Rotating 270°	Any size	12.2, 0.24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32. Horizontal flipping	Any size	14.8, 0.41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33. Vertical flipping	Any size	12.9, 0.31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34. Inserting annotations	512 × 512	28.7, 0.94	0	0	0	0	0.2	0	0	0	0	0	0	0.1/0	0.2/0	0.5	0.7	0.9
Inserting annotations	1024 × 1024	28.4, 0.94	0	0	0	0	0.2	0	0	0	0	0	0	0.1/0	0.2/0	0.7	0.9	1.0
Inserting annotations	1024 × 1024	28.4, 0.94	0	0	0	0.1/0	0.3	0	0	0	0	0	0	0.1/0	0.2/0	0.7	0.9	1.0
35. Darken private data	Any size	Inf, 1.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Watermark-based attack																		
36. $l = 128$	512 × 512	13.0, 0.29	51.5/24.1	24.3/24.2	24.2/18.7	19.0/19.2	19.7	10.2/8.8	9.0/11.2	13.6/14.4	16.2/20.7	20.4/19.8	20.7/20.4	20.9/18.6	20.2/20.5	48.4	47.7	48.5
$l = 128$	1024 × 1024	12.2, 0.27	48.2/25.3	23.3/23.7	24.1/17.6	19.4/19.0	20.3	12.1/10.4	9.9/13.1	13.2/15.6	16.7/19.3	19.9/20.2	21.1/20.5	21.3/21.5	21.5/20.9	53.4	54.2	53.8
$l = 128$	1024 × 1024	12.2, 0.27	50.5/23.6	23.3/23.1	22.2/19.7	20.0/18.2	19.7	14.3/14.5	14.9/15.6	16.7/17.9	18.3/19.8	20.4/20	20.7/20.1	20.9/21.3	20.5/19.7	53.4	54.2	53.8
37. $l = 256$	512 × 512	12.7, 0.28	40.8/36.1	54.7/35.8	36.4/25.4	25.3/24.0	24.4	8.2/8.8	9.2	10.8/11	14.8/17.8	18.8/19.2	19.6/19.3	20.4/20.1	21.4/20.7	47.7	46.9	47.4
$l = 256$	1024 × 1024	12, 0.26	40.6/36.4	52.4/34.8	38.4/27.4	26.3/22.0	23.9	8.9/9.2	9.6/10.3	11/12.4	14.5/15.5	18.5/19.1	19.3/19.4	19.9/19.2	20.2/21.6	49.7	49.9	50.3
$l = 256$	1024 × 1024	12, 0.26	42.8/38.0	53.9/36.8	37.5/26.4	25.7/23.2	23.8	8.2/8.4	9.2/9.5	10.8/13.3	14.8/16.6	18.8/19.2	19.6/20.1	20.4/20	20.9/21.1	49.7	49.9	50.3
38. $l = 512$	512 × 512	12.3, 0.22	51.2/56.3	51.3/55.4	55.6/27.2	27.0/26.9	27.3	10.2/10.4	10.7/11.1	12.7/13	15.4/16.9	17.6/18.9	19.8/20.3	20.8/20.5	21.4/20.9	47.7	47.7	47.8
$l = 512$	1024 × 1024	11.9, 0.23	49.3/54.3	51.5/54.4	53.6/26.1	25.0/24.2	22.3	9.7/10.1	10.4/11.3	12.1/13.1	14.8/16.7	17.1/17.9	18/19.8	20.2/20.3	20.1/20.6	51.3	51.5	51.1
$l = 512$	1024 × 1024	11.9, 0.23	51.2/56.3	51.3/55.4	55.6/27.2	27.0/26.9	27.4	9.1/9.2	9.9/10.8	11.8/12.5	14.9/15.2	16.6/18.8	18.9/19.5	20.3/20.6	20.5/19.8	51.3	51.5	51.1
39. $l = 1024$	512 × 512	12.5, 0.26	43.9/40.8	53.6/41.6	41.5/56.6	57.8/46.3	46.4	11.7/12.1	12.5/13.3	15.4/15.5	16.1/18.6	17.3	19.3/20.3	21.2/21.8	22.6/21.6	50	49.2	50.1
$l = 1024$	1024 × 1024	12.7, 0.28	45.2/41.3	54.3/42	42.7/55.3	58.3/47.5	46.8	12.2/12.8	13.2/14.9	15.5/15.1	16.2/16.8	17/17.7	18.9/20.3	21/20.9	21.3/21.7	49.8	50.2	50
$l = 1024$	1024 × 1024	12.7, 0.28	44.5/41	53.9/41.9	42.2/56.9	58.1/46.7	46.6	11.5/12	12.2/13.1	14.9/15.2	15.8/17.1	16.9/18.5	19.2/20.2	21.3/21.1	20.7/20.3	49.8	50.2	50
40. $l = 2048$	512 × 512	12.3, 0.23	52.1/56.4	53.1/56.6	56.6/57.7	57.5/58.0	58.6	11.3/11.9	12.5/12.4	13.6/16.2	16.7/17.6	18.5/19.8	20.3/21.5	21.9/21	22.3/21.6	48.4	49.2	49.3
$l = 2048$	1024 × 1024	12.3, 0.24	53.3/55.9	54.2/55.2	56/54.4	54.9/54.4	55.6	11.7/11.9	12.8/13.3	14.1/15.9	16.8/18.2	18.3/18.4	20.2/20.8	21.5/22.1	22.1/19.8	50.4	50.2	49.8
$l = 2048$	1024 × 1024	12.3, 0.24	51.7/54.4	52.7/55.5	54.1/55.2	57.1/55.9	56.8	11.9/12.7	13.1/13.9	14.2/16.6	17.3/18	18.8/19.7	20.4/21.7	22.1/21	21.7/22.2	50.4	50.2	49.8

Table 4: Distortion of variable-length stable, semistable and volatile tags when retrieved from an image different from the one they were associated to (see Sections 4.1 and 4.3).

	Distortion of stable tags —NHD (%)—					Distortion of semistable tags —NHD (%)—								Distortion of volatile tags —NHD (%)—		
$Length(T)$ Measure	128/BCH	256/BCH	512/BCH	1024/BCH	2048	128/BCH	256/BCH	512/BCH	1024/BCH	2048/BCH	4096/BCH	8192/BCH	16384/BCH	64	128	256
Mean	48.8	49.2/49.3	49.7	49.8	49.9	49.6	49.9/50	50	50.1	50	50	50	50	50	49.9	50
Standard deviation	4.5/5	3.7/3.6	2.6	1.9	1.3	4/3.9	2.8	2	1.3	1	0.6	0.5	0.4	5	3.5/3.6	2.5
Minimum	13.3/12.5	18.8/19.1	24.2/25.2	30.6	36.7	15.6/15.9	23.6	37.9/37.8	40.7/42.1	44.7/44	46.1/46.3	47/47.1	47.4	31.2/32.3	38.3/34.3	40.4/40

Table 5: Effect of JPEG2000 compression —with compression ratios of 8, 16, 32— on the distortion of stable and semistable tags when the image test set undergoes other common modifications in the medical context. See also the original distortion in Table 3.

	Image size —px^2—	Additive distortion —NHD (%)— of JPEG2000 compression with CR = 8, 16, 32 in stable tags						Additive distortion —NHD (%)— of JPEG2000 compression with CR = 8, 16, 32 in semistable tags																				
$Length(T)$ #. Operation	-	1024-BCH			2048			128-BCH			256-BCH			512-BCH			1024-BCH			2048-BCH			4096-BCH			8192		
Common image processing																												
11. β correction -0.3	512 × 512	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0.4
12. β correction -0.5	512 × 512	0	0	0	0.2	0.1	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.3	0.3	0.6	2.8	2.7	3.2
13. β correction $+0.4$	512 × 512	0	0	0	0	0.1	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0.1	0.1
14. β correction $+0.7$	512 × 512	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.3	0.7	0.8
15. Contrast stretching 2%	512 × 512	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0.1	0.4
16. Contrast stretching 10%	512 × 512	0	0	0	0.1	0.1	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0.3	0.8
17. Invert colors	512 × 512	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18. Local hist. equal.	512 × 512	0	0	0	0.1	0.1	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0.2
Local operations																												
19. Edge sharpening	512 × 512	0	0	0	-0.1	-0.1	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0.1	0	0.2	0.2
20. Median filter 5 × 5	512 × 512	0	0	0	0	0	0	-0.4	-0.5	0.7	-0.4	-0.3	-1	-1.2	-1.1	-2.4	-1.4	-2.1	-2.2	-2.2	-2.4	-3.4	-2.2	-2.2	-4	-2	-2.6	-3.2
21. Median filter 7 × 7	512 × 512	0	0	0	0	-0.1	-0.4	-1.1	-0.9	-1.5	-1.2	-1.3	-1.1	-0.7	-0.5	0.5	-1.4	0.4	-1.4	-0.3	-0.9	-0.5	-0.7	-0.5	-1.6	0.8	0.3	-0.6
22. Image averaging 5 × 5	512 × 512	0	0	0	0	0.1	0.2	-1.2	-1.5	-1.4	-0.8	-1.9	-2.4	-1.8	-1.6	-1.1	-2.4	-3.5	-2.8	-2.4	-2.5	-3.4	-4.6	-2.4	-5	-2.9	-3.9	-4.4
23. Image averaging 7 × 7	512 × 512	0.3	0	-0.3	0.5	0.2	0.3	-1.2	-2.7	-0.5	-2.9	-1.9	-2.5	-1.1	-3.9	-3.3	-3.5	-3.3	-4.9	-2.4	-3.4	-2.6	-2.7	-4.8	-4.8	-2.7	-3.3	-3
24. Gaussian filtering 7 × 7	512 × 512	0	0	0	0.2	0.2	0.2	5.4	5.6	5.4	5.8	5.3	5.4	3.7	6.5	4.2	6.3	4.2	6.3	4.4	5.4	5	4.3	4.6	3.7	5.7	5.6	5.1
25. Gaussian filtering 11 × 11	512 × 512	0	0	0	0.3	0.4	0.2	3.1	1.8	2.8	2	3.1	1.8	4.3	2.2	4.1	1.7	4.3	1.6	4.6	3.9	4.6	3.1	4.8	2.7	3.9	2.8	2.8
26. Motion blur 7	512 × 512	0	0	0	0.3	0.2	0.1	0	0.7	0	0.3	-0.1	-0.6	-0.5	0.2	-0.5	-0.4	-0.8	-0.5	-0.7	0.5	-1	-0.4	0.1	0	-1.8	-1.9	-1.1
27. Motion blur 9	512 × 512	0	0	-0.1	0	0	-0.2	0.1	0.5	0	0.9	-0.2	-0.4	-0.4	0.3	0.7	0.1	0.4	0.4	0.3	0.5	1	0.2	0.2	0	-0.1	0.4	-0.3
34. Inserting annotations	512 × 512	0	0	0	-0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0.2

The results in Table 3 (unshaded cells cells, left side of slashes), demonstrate that the overall robustness of stable tags to any tested image modification is high up to capacities of 512-1024 bits, with an average $NHD < 1\%$, decreasing for aggressive image compression and filtering at 2048 bits. It is worth mentioning that the local operations (except edge sharpening) are the most challenging modifications, since they affect many subbands. In fact, the motion blur modification yields an average $NHD = 3.6\%$ in stable tags of 1024 bits. Since the number of coefficients available from level 3 (aWL for $512 \times 512\, px^2$ images) to the top is 16,384, and only one feature can be extracted from each coefficient, associating tags longer than 2048 bits will make the overall robustness decrease sharply. BCH(511,259,30) coding [61] was successfully tested with stable tags to improve their robustness. Each tag is divided into blocks of 259 bits and encoded as 511-bit redundant blocks. Although it is then possible to correct up to 30 bits per block, the length of the associated keytag is almost double that of the keytag associated to the tag without BCH coding, which reduces its overall robustness. As can be seen in Table 3 (unshaded cells, right side of slashes), the balance is positive and the NHD is reduced to 0 or almost 0 in most cases. Nonetheless, BCH can not be applied to all images for capacities of ≥ 2048 bits, which turn into ≥ 4096 bits after this coding, since some clipped ROIs do not have enough coefficients for the keytag association. Therefore, BCH coding-decoding will be applied to all stable tags with length ≤ 1204 bits, the coding as a previous step to Algorithm 1 and the decoding as a final step after Algorithm 2.

Semistable tags show good robustness to mild image compression and common image processing, e.g. $NHD = 0.2\%$ with L = 4096 for JPEG2000 compression 16:1, $NHD = 0\%$ with $L = 8192$ for $8 : 1$. As was also expected, their robustness to modifications that remove details (e.g. image averaging 7×7) is very low, while for edge sharpening, which enhances them, it is very high ($NHD = 0.6\%$ for $L = 8192$). It is also observed that BCH coding is pertinent for capacities up to 4096 bits, since it reduces the NHD of permissible modifications, which do not affect the clinical value of the image (JPEG2000 CR 16:1, common image processing and edges sharpening). Therefore, semistable tags with length ≤ 4096 bits will implement BCH coding. The NHD of permissible modifications is $\leq 2.1\%$ for capacities of 8192 bits (to be implemented without BCH coding), which is tolerable. Nonetheless, it is not recommended exceeding this capacity since the distortion of permissible modifications increases a lot, e.g. $NHD = 8.7\%$ for JPEG2000 CR 16:1 with tag length of 16384 bits. Finally, volatile tags present very low robustness, with $NHD \approx 50\%$, to any irreversible modification affecting the image ROI even when the tag length is very short.

4.3. Specificity

Specificity is a relevant feature of keytagging, since it measures how much information can be retrieved with a keytag when the image it is associated to is replaced with another image (not derived from the former). The distortion of the tags retrieved from non-original images shall be considerably high for two reasons: to avoid that someone can read the tag content without the original image (thus, affecting its privacy) and to avoid that someone can establish a relation between certain keytag an another image not associated to it (thus, affecting its security). The results of the specificity evaluation, using simulation setup 2 (see Section 4.1), are represented in Table 4. It can be seen that the average values of distortion for any keytag type and length are close to the ideal NHD, 50%, which guarantees perfect destruction of the tag content. Nevertheless, it is also observed that the shorter the keytag, the highest the likelihood of retrieving some tag with lower distortion. In particular, the minimum NHD measured when retrieving the tag content with a very similar image was 12.5% and 15.6% for 128-bit stable and semistable keytags. Although these values far exceed those obtained when evaluating robustness (retrieving the tags from images derived from the original), $NHD = 0\%$ for 128-bit stable keytags and semistable keytags, they shall be taken into account when designing certain keytag-based security measures (see Sections 5.3 and 5.5).

4.4. Effect of JPEG2000 compression

As explained in Section 3.5, the keytagging algorithm has similarities with the JPEG2000 compressor. For this reason, the robustness of stable and semistable tags to JPEG2000 compression is high, as the results in Table 3 demonstrate. Nonetheless, to claim high compatibility with this compressor, the robustness to the image modifications tested in the table must also be evaluated in compressed versions of the original images. Since medical images are expected to be compressed with ratios around 16, we tested with ratios of 8, 16 and 32. The new results, which are compared with those from the uncompressed images, are depicted in Table 5. Positive values imply that the results of the compressed test set are worse, since the NHD increases, while negative values indicate better results for the opposite reason. Thus, it is observed that the effect of keytagging compressed images instead of the original ones is null on the distortion of stable tags if their length is ≤ 512 bits. For 1024 bits, only two filters suffer a slight change, and for 2048 bits the NHD increases by an average of 0.2%, which is not significant. Semistable tags maintain very similar robustness to common processing, with the exception of β correction -0.5 for a high capacity (8192 bits), which becomes significantly worse. Regarding local operations, there is an important change in the results of semistable tags, but their overall robustness to these operations is still low or very low, as intended. Volatile keytags maintain minimum robustness to any modification, with $NHD \simeq 50\%$. Geometrical modifications and darkening private data remain with $NHD = 0$ for stable and semistable tags, and inserting annotations on the compressed images produces no or very slight change. Thus, it can be concluded that keytagging JPEG2000 compressed images instead of the original images obtains very similar results, which permits implementing the same applications with the same operating parameters (see Section 5).

4.5. Average runtime cost

Most of the processes comprising Algorithm 1-2 are of linear complexity, as can be inferred from their description throughout Sections 3.1-3.4. Table 6 shows the runtime cost of these processes when executed in a MATLAB® R2014a implementation running on an Intel Core i5 Quad Core at 2.9 GHz with OS X Yosemite. The slowest process is the tag BCH coding, taking $158 - 187\,ms$, which can be performed offline and is recommended for stable tags with length $\leq$ 1024 bits and for semistable tags with length $\leq$ 4096 bits, and the decoding, which needs to be performed online but takes only $15 - 24\,ms$. The segmentation and the color reduction of the image have a negligible cost, while the wavelet transformation is the second slowest process. Most of its runtime cost is concentrated on calculating the first 3-4 decomposition levels and is highly dependent on the size of the original image. When the size of the image is increased by two in both rows and columns, the runtime cost increases approximately by four. The coding of a keytag has linear complexity and low runtime costs, e.g. $0.1\,ms$ for 128-bit tags and $7.1\,ms$ for 8192-bit tags; and its decoding has a very low fixed cost of 0.3 ms. Regarding cryptographic processes, the digital signature of a keytag and its verification have a low fixed cost, 2.6 and $7.2\,ms$. The cost of encrypting a keytag depends

Table 6: Average runtime cost (in ms, unshaded cells) of the processes for keytag association and tag retrieval depending on different parameter values (shaded cells).

Operation:	Parameter/s	Value 1	Value 2	Value 3	Value 4	Value 5	Value 6	Value 7	Value 8	Value 9	Value 10	Value 11
Keytag association process												
0a. BCH coding of a tag	$length(T)$	128	256	512	1024	2048	4096	-	-	-	-	-
		157.9	162.0	188.1	185.3	185.7	187.2					
1a. Segmentation, color reduction and wavelet transformation of I_{or}, Algorithm 1: lines 2-5	WL Image size	1	2	3	4	5	6	7	8	9	10	11
	$512 \times 512\, px^2$	19.9	21.4	23.9	25.3	25.4	25.4	25.6	25.6	25.7		
	$1024 \times 1024\, px^2$	87.6	95.9	105.0	110.4	110.4	110.6	110.8	110.9	110.8	111.1	
	$2048 \times 2048\, px^2$	359.7	421.7	455.2	479.0	479.3	478.1	478.4	478.5	478.4	478.7	478.7
2a. Coding of a keytag, Algorithm 1: lines 6-7, 9-12	$length(T)$	64	128	256	512	1024	2048	4096	8192	-	-	-
		0.1	0.1	0.2	0.5	0.9	1.9	3.6	7.1			
3a. Signature of a keytag, Algorithm 1: line 13							2.6					
4a. Encryption of a keytag, Algorithm 1: lines 14-15	$length(T)$	64	128	256	512	1024	2048	4096	8192	-	-	-
		0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2			
5a. Encryption of a user's access keys, Algorithm 1: lines 17-21							0.45					
Tag retrieval process												
1r. Segmentation, color reduction and wavelet transformation of I_{or}, Algorithm 2: lines 2-5	WL Image size	1	2	3	4	5	6	7	8	9	10	11
	$512 \times 512\, px^2$	19.9	21.4	23.9	25.3	25.4	25.4	25.6	25.6	25.7		
	$1024 \times 1024\, px^2$	87.6	95.9	105.0	110.4	110.3	110.6	110.8	110.9	110.8	111.1	
	$2048 \times 2048\, px^2$	359.7	421.7	455.2	479.0	479.3	478.1	478.4	478.5	478.4	478.7	478.7
2r. Decryption of a user's access keys, Algorithm 2: line 7							5.3					
3r. Decryption of a keytag, Algorithm 2: line 9	$length(T)$	64	128	256	512	1024	2048	4096	8192	-	-	-
		0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2			
4r. Verification of keytag signature, Algorithm 2: line 10							7.2					
5r. Decoding of a keytag, Algorithm 2, lines 6, 11-15							0.3					
6r. BCH decoding of a tag	$length(T)$	128	256	512	1024	2048	4096	-	-	-	-	-
		15.3	15.8	15.9	16.7	19	23.8					

linearly on the length of the tag and its runtime cost is very low, 0.2 ms for a 8192-bit tag. The costs of encrypting and decrypting a package of up to 16 access keys to keytags intended for a user are 0.45 and 5.3 ms.

According to the data in Table 6, the overall delays for associating several keytags (in this example 4 128-bit stable, 3 2048-bit semistable and 1 256-bit volatile) to an $512 \times 512\, px^2$ image (operations 1a-3a) and retrieving the corresponding tags (operations 1r, 4r-6r) are $< 55\, ms$ and $< 115\, ms$ respectively. If keytagging is integrated within the JPEG2000 compressor, these overall delays drop to $\leq 30, 90\, ms$ for any image size, since this compressor performs the wavelet transformation. Finally, when some tags are private, it is necessary to encrypt and decrypt (operations 4a and 3r) the keytags and the corresponding access keys (operations 5a and 2r) of each user. These operations add an extra delay of $< 0.5 \cdot \#users\, ms$ in the keytag association and 5.5 ms in the tag retrieval procedure.

4.6. Scalability

The scalability of any technique for protecting medical images is an important feature since the tendency is to increase their resolution for better image processing and visualization. The shaded cells in Table 3 depict the robustness-capacity trade-off in a second image test set. This corresponds to the original data set introduced in Section 4.1 after bicubic interpolation by a factor of two in both rows and columns, which has been processed with the same modifications as the former but tuning the parameters so that the image distortion caused is closely similar in $PSNR$ and $MSSIM$. Otherwise, if the parameter values for the image modifications are maintained and the images are enlarged, the distortion caused is usually lower (especially when the image undergoes compression and local operations) and the robustness-capacity tradeoff would improve only because the image is less degraded.

Two different parameter configurations have been tested, using the aWL in Algorithm 3: $extractFeatures$ (the lightly shaded cells in Table 3) and reusing the same aWL as for the original test set, comprised by $512 \times 512\,px^2$ images (the shaded cells in Table 3). When associating stable keytags and performing image compression and common image modifications, it can be seen that both options obtain results similar to those for $512 \times 512\,px^2$ images up to tag lengths of 1024 bits. The second option gives better results, but these are still not good enough to allow capacities higher than 1024 bits. When associating stable keytags and performing local image operations, the first option obtains much better results than the second, since the tag is associated to lower frequencies of the image which better endure these types of image modifications. These results are very similar to those obtained with $512 \times 512\,px^2$ images. Regarding semistable keytags, the first option also obtains results that are closer to those obtained with $512 \times 512 px^2$ images. The configuration for volatile keytags has been maintained by selecting coefficients from the HH subband in the first decomposition level, to guarantee very low robustness. Regarding the rest of the image modifications, the results are equal or very similar for both options. As a general conclusion, the parameters proposed in Algorithms 1-2 guarantee the scalability of keytagging with respect to the robustness-capacity trade-off, which is maintained for different image sizes.

Regarding the runtime cost of keytagging, increasing the size of the image increases the cost of performing its wavelet transformation (operations 1a and 4r in Table 6) by an n^2 factor. For instance, calculating the MWL of a $512 \times 512 px^2$ image takes $25.7\,ms$, while for a $1024 \times 1024 px^2$ image it takes $111.1\,ms$ and for a $2048 \times 2048 px^2$ image it takes $478.7\,ms$. As a result, the overall delays for associating keytags and retrieving tags presented in Section 4.5 ($\simeq$ 55, 115 ms) increase to $\simeq$ 140, 200 ms for $1024 \times 1024\,px^2$ images and to $\simeq$ 510, 570 ms for $2048 \times 2048\,px^2$. Therefore, it is recommended combining keytagging with JPEG2000 compression in order to guarantee the scalability of keytagging, since it reduces these delays to $\simeq 30, 90\,ms$ for any image size.

5. Protection of the medical image by means of keytagging

The integration of keytagging to strengthen the protection of images transmitted within medical architectures is analyzed in Section 5.1. In addition to this, the operating parameters of keytagging are adjusted

Table 7: Recommended parameters to implement different keytag-based security measures (see Section 5)

Security measure	**Keytag type**	***aWL*, allowed wavelet level/s**	**Tag content**	**Tag length** (bits)	**Tag BCH coding**	***Th*, tag detection threshold** (bits)	**Keytag encryption**	**Issuer of DS keytag**
Image resynchronization	Stable	$\geq MWL(ROI_g) - 6$	Reference	128	Yes	Most similar/ dissimilar tag	No	Image acquisition device or image issuer
Authentication and traceability	Stable	$\geq MWL(ROI_g) - 6$	Reference	128	Yes	126	No	Image acquisition device or image issuer, and each entity that processes the image
Pursuing illegal image copies	Stable	$\geq MWL(ROI_g) - 6$	ID of image copyright holder	128	Yes	126	No	Holder of image copyright
Image purchase	Stable	$\geq MWL(ROI_g) - 6$	ID of image buyer	128	Yes	126	Yes	Holder of image copyright
Private captioning with RBAC	Semistable	$\leq MWL(ROI_g) - 7$	Text/codes	≥ 256 ≤ 8192	No	-	Yes (RBAC)	User authorised to update test data
Integrity control	Volatile	1 (HH subband)	Reference	64	No	-	No	Image acquisition device or image issuer
Location of tampered areas	Volatile	1 (HH subband)	Reference	≥ 512	No	-	No	Image acquisition device or image issuer

in order to implement the security measures proposed in Section 1. The use of specific keytags for image resynchronization, authentication and traceability, copyright protection, private captioning, integrity control and location of tampered areas is summarized in Table 7, and described in detail throughout Sections 5.2-5.7. Finally the security of the keytagging method, including all the security measures that it can implement, is comprehensively assessed in Section 5.8.

5.1. Integration of keytagging in medical architectures

The keytagging method is aimed for supplementing the security in the transmission of images within cooperative medical architectures, which at a technical level are typically supported by standards like DICOM (see Section 6). As portrayed in Figure 3, the integration of keytagging for the sharing of information among different users in these architectures poses three constraints:

- Although the keytagging algorithm guarantees that keytags are protected and verified (by means of cryptography, see Section 3.4), the integration of keytagging in medical architectures requires the implementation of a reliable authorisation mechanism to control which specific entities (e.g. the device that acquired the image the keytag is associated to, a physician in charge of supervising the image, the holder of the image copyright, etc.) are entitled to add, remove or consult keytags of an image.
- The medical architecture shall also implement a mechanism to record which keytags are added, removed or consulted, in order to guarantee the auditability of these events and the accountability of the entities involved.

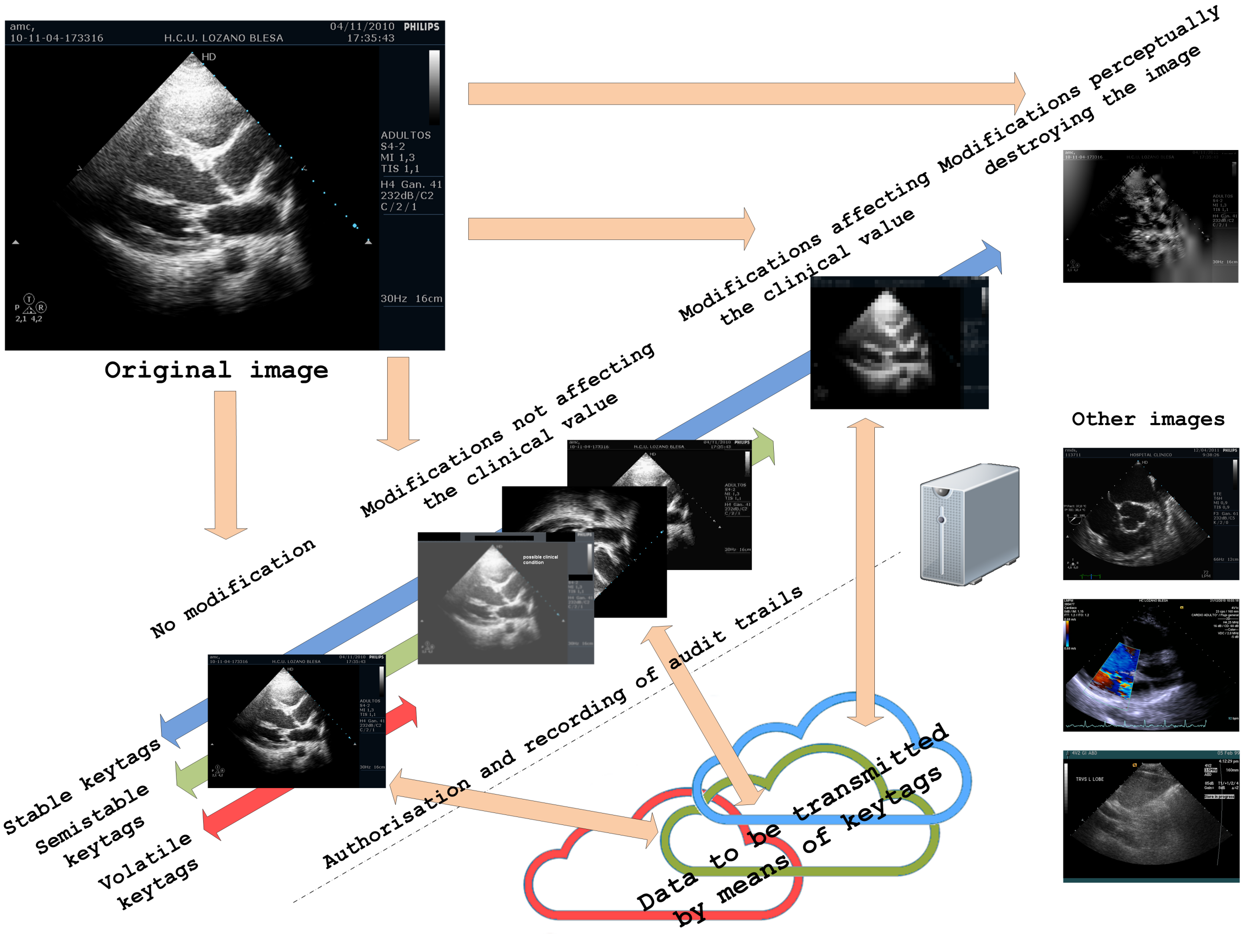

Figure 3: Overview of a medical architecture integrating keytagging.

- The capacity of the different types of keytags to transmit information among different users is conditioned by the perceptual and clinical distortion that their instances of the image —associated to the keytags— may undergo. This property is used for the implementation of the keytag-based security measures described throughout Sections 5.2-5.7.

5.2. Image resynchronization

The medical image may be subject to geometrical transformations, which in the medical context may be 90/180/270° rotation, vertical or horizontal flipping. Being able to find the original position of the image is essential to retrieve its tags correctly, since otherwise they would be desynchronized. To accomplish this task, a reference synchronization tag is associated to the image. This is a 128-bit public stable tag, which is retrieved six times, from the received position and from each of the geometrical transformations (by rotating/mirroring each subband in $Coef$ just after line 5 in Algorithm 2). The retrieved tag which is most similar to the original reference corresponds to the transformation that returns the image to its original position. If it is observed that a retrieved tag is more dissimilar than the most similar tag, e.g. 127 wrong bits in the former and 80 correct bits in the latter, this means that the original position corresponds to the former, whose colors have been inverted. In that case, the values of $Coef$ need to be inverted (just after line 5 in Algorithm 2).

5.3. Image authentication and traceability

Authentication refers to the capacity to determine if an image is either derived from another, including perfect copies in this category, or if it is unrelated. To implement image authentication, a reference public stable tag is associated to the original image. It is then retrieved from the image to be authenticated. Only if the reference and retrieved tags are very similar or equal is the image positively authenticated. The authentication ability of keytagging is adjusted by means of two operating parameters: the length of the reference tag and the detection threshold, Th. Assuming that decoding each tag bit has a probability of success of $\simeq 0.5$ when it is done from a wrong image, the probability of false positives P_{fp} in authentication can be approximated by means of the following binomial distribution:

$$P_{fp} = \sum_{i=Th}^{n=length(T)} (0.5)^n \cdot \frac{n!}{i!(n-i)!} \tag{4}$$

The image is positively authenticated only if the number of correctly decoded tag bits exceeds the threshold, $\sum_{i=1}^{length(T)} (\tilde{T_i} = T_i) \geq Th$. On the one hand, the probability of false positives P_{fp} in image authentication is expected to be similar to that required in applications of biometric recognition, $\leq 10^{-6}$. Thus, avoiding images that were not keytagged for authentication from obtaining false positives requires setting a high Th. On the other hand, too high values of Th (low permitted NHD) will increase the probability of false negatives (non-authenticated images that were actually associated with the authentication tag). As can be seen in Table 3, the robustness of stable tags with BCH coding is total ($NHD = 0\%$) for tag lengths up to 512 bits, which ensures no false negatives. However, a shorter tag is enough to ensure that the likelihood of false positives, P_{fp}, is very low. Using $length(T) = 128$ and $Th = 126$ ($NHD = 1.6\%$) makes $P_{fp} = 2.5 \cdot 10^{-35}$. The minimum NHD in simulation setup 2 (see Section 4.3) was obtained when associating-retrieving a reference 128-bit tag using the two most similar images in the test set ($PSNR = 23dB$), two close slices from a PET-CT. That NHD value, the closest to causing a false positive, was 12.5%, still far greater than Th ($= 1.6\%$). To sum up, these operating parameters ensure the perfect authentication of the whole test set.

Traceability policies intend to facilitate the tracking of entities that process or simply forward the medical image test. This can be easily accomplished if each entity validates the digital signature/s of the authentication keytag, authenticates the image and adds its own digital signature to the authentication keytag if the previous verifications are positive. Otherwise, the image is reported as replaced or heavily tampered with and requested from the last entity that validated it.

5.4. Copyright protection

Copyright protection may be implemented by means of a specific double authentication mechanism (see Section 5.3), involving the image copyright holder and each image buyer, in the following manner:

- The copyright holder is identified by means of a public ID (e.g. Rubio@eHealthZ14), which he/she associates to the image by means of a 128-bit stable tag. Besides, this tag has a secondary purpose: enabling the automatic search for illegal copies of the image in internet sites and databases. An internet bot may use the keytag to retrieve tags from the images in targeted sites and compare them with the copyright holder ID. If some image is positively authenticated, it is an illegal copy.

- Each buyer is identified by means of an ID, which the copyright holder associates to the image with a 128-bit private stable tag. In this manner, any buyer can prove that he/she holds a legal copy, even if he/she has made substantial modifications to it.

5.5. Private captioning with RBAC

The association of private information with the image, only retrievable by authorised users if the image preserves its clinical value, may be easily carried out by means of semistable private tags. Nonetheless, the results in Table 3 suggest that the overall length of these tags should not exceed 8192 bits. Therefore, it is recommended compressing them as much as possible, e.g. by replacing text with codes. Nevertheless, it is also recommended that the overall tag length is not too short, to guarantee a good specificity. As pointed out in Table 4, a minimum length of 256-bit ensures that the minimum distortion (NHD) of tags retrieved from very similar images (but not derived from the original) is high, $> 23\%$. Therefore, short tags shall add padding bits until their length is ≥ 256 bits.

Cryptographic-based RBAC may be applied to improve private captioning (see Algorithms 1: lines 14-21 and Algorithm 2: lines 7-9). For each tag, its associated keytag is symmetrically encrypted with a specific symmetric key, Sk, and all the symmetric keys corresponding to tags intended for a user are encrypted with his/her public key, $PbUser\{i\}$. Thus, each user decrypts his/her symmetric keys $\{Sk\}$ with his/her private key, $PrUser\{i\}$, and then decrypts his/her keytags with these symmetric keys. All this can be easily managed by encapsulating the keytags with CMS. There are two reasons to implement RBAC in this manner, instead of by associating a different keytag for each user. First, because all the users retrieve the same tag content, even when the image is modified. Otherwise the users may retrieve tag contents with different degrees of distortion. Second, because associating several keytags with the same content would reduce the overall capacity.

5.6. Integrity control and location of tampered areas

The detection and location of modifications affecting the image may be efficiently performed by means of public volatile tags. Table 3 shows that these tags suffer high distortion even when the image modifications are mild, which implies very low robustness. Given these results, we propose the following:

- For integrity control, it is sufficient to use 64-bit reference tags, since they have an average $NHD \simeq 50\%$. Thus, approximately 32 bits are wrongly detected when the image ROI undergoes non-geometrical modifications (geometrical modifications are reversed by means of resynchronization, see Section 5.2). The exception to this, NHD slightly > 0, occurs when the image is partially annotated in the ROI, since only a few pixels in isolated regions change.

- For the location of tampered areas, the position in BM of wrongly detected tag bits is marked in the corresponding positions of the image. The even distribution of tampered pixels, detected after a

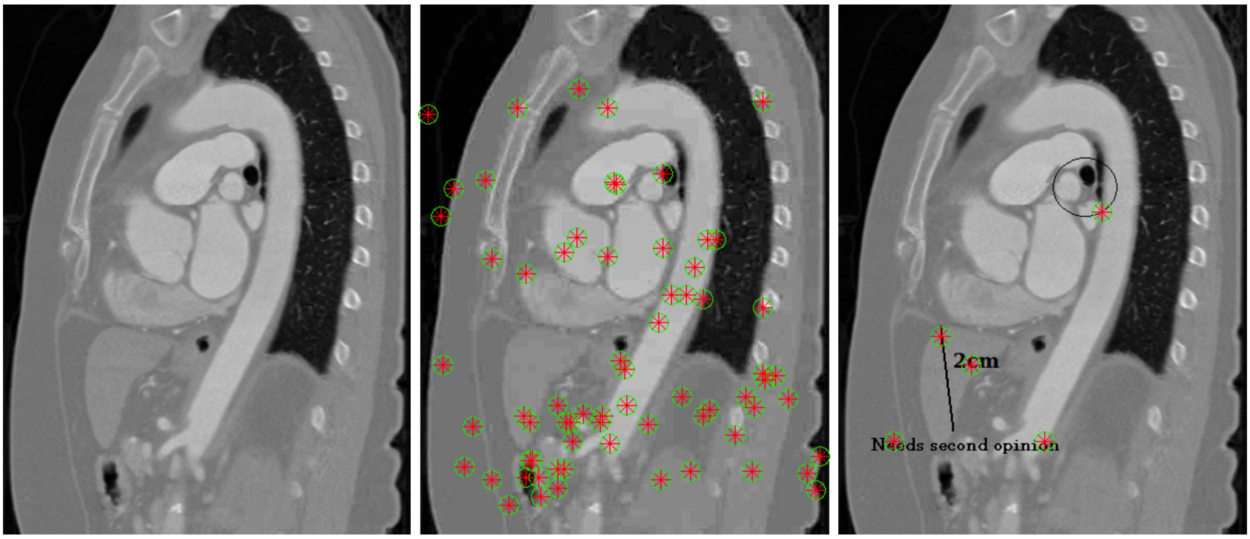

Figure 4: Location of tampered areas on the ROI of a $512 \times 512\,px^2$ original image (left), on its JPEG $QF = 15\%$ compressed version (center, tag length = 128 bits) and on an annotated version (right, tag length = 512 bits).

common image modification, is shown in Figure 4: center. Logically, longer reference tags are able to achieve finer granularity in the delimitation of modified image areas, which is especially important in the case of detecting annotations in the image ROI. Considering that usually only $\simeq 2\%$ pixels have been annotated and half (1%) change the value of the features used the for coding of these keytags (the LSB of certain wavelet coefficients), a 512-bit tag is able to locate approximately 5 tampered areas in the ROI, as shown in Figure 4: right.

It is worth noting that implementing location of tampered areas already ensures integrity control, but not conversely.

5.7. Simultaneous implementation

This section analyzes whether all the previous security measures can be implemented simultaneously in the images from the test set, by analyzing the results of robustness-capacity in Table 3 and the keytagging parameters in Table 7. According to these results, the security measures based on public stable tags can reach an overall capacity of 512 bits with $NHD = 0$, and they only require 384 bits. Regarding semistable tags, their capacity shall be adjusted to ≤ 8192 bits in order to maintain an adequate robustness in private captioning. With respect to volatile tags, there are no capacity restrictions. Thus, all these security measures can be implemented simultaneously. Furthermore, in this case, it is recommended that the tags for image resynchronization, authentication-traceability and location of tampered areas-integrity control associate the same reference. This would require extending the reference used for location of tampered areas, e.g. by repeating several times the shorter reference used to implement the other security measures. In this manner, the reference does not need to be transmitted since comparisons may be established among the tags retrieved. Consequently, the keytagging-based security system would be able to operate in a blind manner.

An overview of the overall keytagging system is introduced below by means of the following use case. A patient's medical image is acquired by means of a CT scanner, whose software generates a 128-bit reference and runs the keytagging algorithm to associate it by means of three keytags. Two of these keytags are stable, intended for resynchronization and authentication, and the third is volatile (resulting from the concatenation of the reference 8 times), intended for the location of tampered areas. The keytags are attached with the image file and recorded in the audit trail system of the medical architecture to which the CT scan is connected. The access control system of the architecture establishes that this image can be edited by two specialists of the patient and consulted by his general practitioner. Each time that one of them accesses the image, the visualization software runs the keytagging algorithm to validate the keytags associated and read their content. Next, the reference introduced after the acquisition is used to authenticate and resynchronize the image (if necessary), and also to pinpoint tampered areas (if any). The specialists can introduce text regarding the diagnosis of the patient or his/her treatment, which the visualization software will associate by running the keytagging algorithm to add semistable keytags. These keytags will be attached in the image file and recorded in the audit trail system. The three users can consult the image, perform modifications on it and, using the visualization software, consult the keytags. In spite of possible modifications, the visualization software will still be able to authenticate and resynchronize the image, to display the content of the semistable keytags (if the image has not lost its clinical value) and to pinpoint the modified areas. Finally, the specialists (but not the practitioner) can use the visualization software to order the removal of their own semistable keytags from the image file, being this event recorded in the audit trail system.

5.8. Risk assessment

The keytagging method, described in Section 3, involves different elements; namely, the keytagging algorithm itself, the keytagged image, the keytag/s associated to it and the content of the corresponding tag/s. Several considerations can be done about the character, either public or private, of these elements. In the first place, Kerckhoff's principle states that the system shall be secure even if everything about it, except certain keys, is public knowledge. Hence, considering that the enemy will eventually discover the keytagging algorithm [62], it is proposed in Section 3.4 to make it public from the beginning. Similarly, medical keytagged images cannot be considered as impossible to obtain under any circumstance. In fact, certain situations may facilitate attackers to obtain an image copy. Some patients may give their informed consent to the use of their medical images for certain purposes —e.g. teaching, research— after anonymization, which increases the number of accesses to the image (and the number of potential opportunities for attacker accordingly); and even strictly confidential images may be a reasonably target for attackers if at a certain time they are handled (e.g. filtered, annotated) out of a protected standardized format (e.g. as JPX instead of DICOM files). Regarding keytags and their associated tag contents, they can be either public or private depending on the security measures that they implement (see Table 7). The former are available to anyone for consultation, while the latter are considered as the most difficult elements to be obtained (in clear) by an attacker.

With the purpose of weakening the security of the system, an attacker with access to some of its elements may try to perform certain attack/s to interfere with the security measures described throughout Sections 5.2-5.6. Therefore, performing a comprehensive risk assessment, comprising all feasible attacks and the existing countermeasures, is essential for the prevention of potential security breaches. The following risk assessment, based on reference publications on watermarking security [63, 64, 65] and tailored to the specifics of keytagging, analyzes attacks depending on the keytag-based measures affected, the actions intended by the attacker and the system elements he/she needs access to. The following attacks may potentially affect the privacy in image purchase and in image captioning (see Sections 5.4-5.5):

- Unauthorised detection and reading of private keytags. It is impossible to detect keytags if the attacker only knows the image: no information can be extracted from the image alone since keytagging does not modify it in any manner. As regards to reading the whole content of private tag/s associated to an image, the algorithm requires both the image and the plain keytag/s associated. With respect to the former, the attacker may try to use another image if he/she does not have the original, but the specificity of keytags guarantees that the content retrieved will be highly distorted, as demonstrated in Section 4.3. Regarding the latter, keytag/s is/are protected with adequate encryption (see Section 3.4), which makes obtaining its/their plain version/s very unlikely. There is also another possible attack, which —generally speaking— requires even more knowledge about the system and only permits reading part of certain tag/s content. To explain this attack, it is worth reminding that while public keytags shall be associated to different features of the image to avoid eavesdropping, different independent users may associate private keytags to certain repeated image features. Therefore, if an attacker knows one or several plain private keytags with its/their associated tags, he/she would be able to read those tag bits from other keytags associated to the same image features. Nevertheless, this requires the attacker being able to break the encryption of the keytags from different users (to obtain the plain versions) and to know the tag content of at least a private keytag, which is highly unlikely.

And these attacks may potentially affect all the security measures described throughout Sections 5.2-5.6:

- Writing of forged keytags. The attempt to copy a legitimate keytag in another image will not succeed since the keytagging algorithm guarantees that keytags are dependent on the image. This high specificity guarantees that the tag content read will be highly distorted, as proved in Section 4.3. Alternatively, any attacker can decide to associate his/her own keytag/s to any image, since the keytagging algorithm is intended to become public. Nevertheless, the attacker cannot add the required digital signature of a trusted entity to the keytags, unless he/she has broken or stolen the private key of a trusted entity —which is highly unlikely.
- Malicious removal of legitimate keytags. There are three possibilities: attacking the keytag, the image it is associated to or the association. As regards to keytags, although they may be attacked, its integrity and provenance can be evaluated any time by means of its digital signature, which is a mandatory element (see Table 7). As explained before, to forge the digital signature of the signatory entity, so that the attack cannot be detected, the attacker needs to break/steal his/her private key. Regarding attacks on the image, their effects on the robustness of the different types of keytags have been evaluated in Section 4.2. In fact, the different measures described throughout Section 5.2-5.6 are designed based on the intended robustness of keytags —e.g. authentication is based on stable keytags, which can be detected even when the image undergoes important modifications; tamper detection is based on volatile keytags, which get highly distorted even if the modification/s on the image are minor. Therefore, the effect of the modifications of the image have already been taken into account in the design of the security measures. Alternatively, the keytagging algorithm can be exploited to change the value of image features used to encode the keytags. In point of fact, this attack can destroy the tag content when the plain keytag is known (e.g. if it is public), but at the cost of destroying the image as well (see Section 4.2). Regarding collusion attacks (popular in watermarking), which combine different keytagged versions of an image to produce another image with distorted keytags, they would be pointless since any keytagged version of an image is exactly like the original. Finally, the attacker may attempt to make the tag reader think that certain keytags are associated to a different image, with the intention of confusing him/her. To detect this attack, it has been proposed associating a reference tag with two independent stable keytags (one for image resynchronization and another for

image authentication and traceability, see Section 5.7). In this manner, the tag content of these keytags only match when retrieved from the original image (or from an image derived from it, e.g. compressed or filtered).

- Malicious edition of legitimate keytags. This is a combination of the previous types of attack. Therefore, it requires knowing the plain keytag (breaking its encryption if it is private), editing the content of the keytag total or partially (as intended by the attacker), replacing the previous signature with a new valid one (which requires the private key of the original keytag signatory) and, if the keytag is private, re-encrypting the keytag with the same cryptographic key used for its decryption.

The careful design of the keytagging algorithm, its combination with adequate cryptographic elements and the choice of the parameters to enforce the different security measures (mainly the robustness of the keytags) are the foundations of the keytagging method. From this security assessment, it can be concluded that the only manner to weaken the security of this system is by attacking the cryptographic protection of the keytags. Hence, the security of keytagging cannot be considered as lower than the security of cryptography. On the contrary, the former requires an additional element for the retrieval of the content associated: the image.

6. Integration with DICOM

In the present e-health model, the access of users to medical images is controlled by the Health and Radiology Information Systems (HIS and RIS) of the hospital, which communicate with the Picture Archive and Communication System (PACS) where they are stored and managed. In turn, the PACS communicates with the devices in charge of the acquisition to obtain the medical images. All the processes involved in this context, including storage, transmission, handling and impression of medical images are usually regulated by the DICOM [1] standard, whose robust file format and network protocol are currently supported by the vast majority of vendors of medical equipment. To fulfill legal regulations, the security of the DICOM file is very high and it also implements a reliable role-based access control to protect patient privacy. The most sensitive DICOM contents, its confidential tags and the image (if it helps to identify the patient), are put into digital envelopes and sealed by means of CMS. Nonetheless, this purely cryptographic policy has some disadvantages. The first major issue is that the content of a non-tampered envelope is technically retrievable even if the rest of the file has been corrupted. It would be preferable if those DICOM tags containing the most sensitive information were to become unreadable if the image is corrupted. Furthermore, it would be desirable that the image could not be requested if it has been seriously distorted or replaced. The second major issue, explained in Section 1, is the difficulty of developing cooperative architectures that allow authorised users to update the DICOM file and share it with the rest of the users while maintaining its integrity control, authenticity and traceability.

Preserving the DICOM digital envelopes, which already implement CMS, different keytags may be used to associate DICOM tags to the image by replacing the original values with those of its keytags. The DICOM data model is defined by attributes that identify information entities, namely patient, study, series and instance of the image. Since the patient ID tag (0010,0020) is the most important, it would be interesting to associate it to the image permanently. Thus, our proposal is to associate the patient ID by means of two stable keytags, one used for authentication and traceability, and the other for resynchronization. Besides, concatenated four times to lengthen, it may also be associated by means of a volatile keytag for integrity control and location of tampered areas. Finally, semistable keytags may be used to associate important private DICOM tags, intended to become uninterpretable if the image loses its clinical value — e.g. due to aggressive image processing. These tags might be some of those included in the DICOM Basic Diagnostic Imaging Report, Study Date tag (0008,0020), Accession Number tag (0008,0050), Modality tag (0008,0060), Manufacturer tag (0008,0070), Referring Physician's Name tag (0008,0090), Responsible Organization tag (0008,0116), etc.

The features of keytagging, non-modification of the image, low complexity (shown in Section 4.5), scalability (shown in Section 4.6), compatibility with JPEG2000 to encode the image (explained in Section 3.5) and with CMS to protect the keytags, both implemented by DICOM, all together guarantee that the proposed stable, semistable and volatile keytags can be seamlessly integrated within this medical imaging standard to enhance its security and privacy.

7. Conclusions and future work

Keytagging, a method based on associating tags of variable robustness to the image ROI, has proved to be an interesting alternative for the protection of medical image-based tests. In fact, this technique has a series of advantages over watermarking and cryptography alone. It preserves the image clinical content, prevents image forgery and collusion attacks, permits associating data to very stable features of the image and obtains a substantial capacity-robustness tradeoff by means of simple operations. In addition, the keytags (keys to access the tags) are tightly bound to the image, since no tag bit can be derived from them alone or by using an unrelated image. Tested with 512×512 and 1024×1024 pixel images, the robustness of stable tags to modifications that are typical in the medical context is total ($NHD = 0$) up to 512 bits and still very high (only three modifications produce $NHD > 0$) for 1024 bits. Semistable tags obtain good robustness to JPEG2000 CR 16:1 image compression and contrast and brightness change ($NHD = 0\%$ for tag length of 4096 bits), and bad, as intended, to local operations that distort the details of the image (average $NHD > 15\%$ for tag length ≥ 512 bits). Volatile tags achieve very little robustness to any modification of the image, even when associating very short tags ($NHD > 45\%$ for length ≥ 64 bits). Regarding runtime costs, associating a set of keytags and retrieving the corresponding tags for the simultaneous implementation of complementary security measures takes $\simeq 55$, $115\,ms$ for $512 \times 512\,px^2$ images; $\simeq 140$, $200\,ms$ for $1024 \times 1024\,px^2$ images and $\simeq 510$, 570 for $2048 \times 2048\,px^2$ images. To guarantee the scalability of this approach, it has also been demonstrated that this method can be combined with JPEG2000 compression, maintaining its robustness while reducing the keytag association-tag retrieval delays to only $\simeq 30$, $90\,ms$ for any image size.

As future work, we will evaluate improved keytag encoding methods in order to reduce the overhead, which is the main drawback of keytagging with respect to watermarking and cryptography. In addition to this, we intend to avoid the need for BCH coding in stable and semistable tags and to raise the capacity of semistable tags. To achieve this, we will research into the combined use of both most significant and sign bits of high magnitude coefficients as stable features, and also on the use of new transforms that may perform better than wavelets.

To sum up, keytagging achieves the transparent, secure and efficient association of stable, semistable and volatile tags to medical images. This enables the simultaneous implementation of private captioning with role-based access control, integrity control and location of tampered areas, authentication, traceability and copyright protection. Finally, it has been explained how this method can be seamlessly integrated within DICOM to facilitate the deployment of efficient medical architectures where different authorised users can update DICOM files with new information and share it without sacrificing any security measure.

Acknowledgements

This research work has been supported by project TIN-2011-23792/TSI from the Ministerio de Economía y Competitividad (MINECO), the European Regional Development Fund (ERDF), Gobierno de Aragón (research group T98) and the European Social Fund (ESF).

References

[1] DICOM, Digital Imaging and COmmunications in Medicine, National Electrical Manufacturers Association (NEMA), 1993. Accessed in February 2015, `http://goo.gl/BtYbVP`.

[2] G. Eysenbach, What is e-health?, Journal of Medical Internet Research 3 (2001) e20.

[3] HIPPA, The Health Insurance Portability and Accountability Act (P.L.104-191), 1996. Enacted by the U.S. Congress.

[4] PIPEDA, The Personal Information Protection and Electronic Document Act, 2000. Enacted in Canada for protection of health information against commercial use.

[5] LOPD, Ley Orgánica de Protección de Datos de carácter personal (Data Protection Act), 1999. Enacted in Spain.

[6] I. J. Cox, M. Miller, J. Bloom, Watermarking applications and their properties, in: Proceedings of the International Conference on Information Coding and Computing (2000), pp. 6–10.

[7] G. Coatrieux, L. Lecornu, B. Sankur, C. Roux, A review of image watermarking applications in healthcare, in: 28th IEEE Annual International Conference on Engineering in Medicine and Biology Society, EMBS 2006, pp. 4691–4694.

[8] R. Housley, Cryptographic Message Syntax (CMS), RFC 5652, IETF Network Working Group. Accessed in February 2015, `http://tools.ietf.org/html/rfc5652`, September 2009.

[9] L. O. M. Kobayashi, S. S. Furuie, P. S. L. M. Barreto, Providing integrity and authenticity in DICOM images: a novel approach, IEEE Transactions on Information Technology in Biomedicine 13 (2009) 582–589.

[10] G. Coatrieux, H. Maitre, B. Sankur, Y. Rolland, R. Collorec, Relevance of watermarking in medical imaging, in: Proceedings of the IEEE International Conference on Information Technology Applications in Biomedicine (2000), pp. 250–255.

[11] D. S. Taubman, Author, M. W. Marcellin, Editor, M. Rabbani, Reviewer, JPEG2000: Image Compression Fundamentals, Standards and Practice, Journal of Electronic Imaging 11 (2002) 286–287.

[12] D. Osborne, D. Abbott, M. Sorell, D. Rogers, Multiple embedding using robust watermarks for wireless medical images, in: Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia, MUM '04, pp. 245–250.

[13] F. Y. Shih, Y.-T. Wu, Robust watermarking and compression for medical images based on genetic algorithms, Information Sciences 175 (2005) 200–216. Medical Image Processing.

[14] X. Zhou, X. Duan, D. Wang, A semifragile watermark scheme for image authentication, in: Proceedings of the 10th International Multimedia Modelling Conference, pp. 374–377.

[15] W. Gang, R. Ni-ni, A fragile watermarking scheme for medical image, in: 27th Annual International Conference of the Engineering in Medicine and Biology Society, pp. 3406–3409.

[16] I. Kallel, M. Kallel, M. Bouhlel, A secure fragile watermarking algorithm for medical image authentication in the DCT domain, in: Information and Communication Technologies, 2006. 2nd ICTTA '06., volume 1, pp. 2024–2029.

[17] X. Zhou, S. A. Lou, H. Huang, Authenticity and integrity of digital mammographic images, in: Medical Imaging'99, International Society for Optics and Photonics, pp. 138–144.

[18] S.-G. Miaou, C.-M. Hsu, Y.-S. Tsai, H.-M. Chao, A secure data hiding technique with heterogeneous data-combining capability for electronic patient records, in: Proceedings of the 22nd IEEE Annual International Conference on Engineering in Medicine and Biology Society (2000), volume 1, pp. 280–283.

[19] H.-M. Chao, C.-M. Hsu, S.-G. Miaou, A data-hiding technique with authentication, integration, and confidentiality for electronic patient records, IEEE Transactions on Information Technology in Biomedicine 6 (2002) 46–53.

[20] A. Giakoumaki, S. Pavlopoulos, D. Koutouris, A medical image watermarking scheme based on wavelet transform, in:

Proceedings of the 25th IEEE Annual International Conference on Engineering in Medicine and Biology Society (2003), volume 1, pp. 856–859.

[21] A. Giakoumaki, S. Pavlopoulos, D. Koutsouris, Secure and efficient health data management through multiple watermarking on medical images, Medical and Biological Engineering and Computing 44 (2006) 619–631.

[22] S.-H. Yang, W.-J. Liao, A compressed domain image watermarking scheme with the SPIHT coding, Journal of Information Science and Engineering 26 (2010) 1755–1770.

[23] O. J. Rubio, A. Alesanco, J. Garca, Secure information embedding into 1D biomedical signals based on SPIHT, Journal of Biomedical Informatics 46 (2013) 653–664.

[24] A. Wakatani, Digital watermarking for ROI medical images by using compressed signature image, in: Proceedings of the 35th Annual Hawaii International Conference on System Sciences, HICSS 2002, pp. 2043–2048.

[25] C.-S. Woo, J. Du, B. L. Pham, Multiple watermark method for privacy control and tamper detection in medical images, in: Proceedings of the APRS Workshop on Digital Image Computing, pp. 59–64.

[26] H.-K. Lee, H.-J. Kim, S.-G. Kwon, J.-K. Lee, ROI medical image watermarking using DWT and bit-plane, in: Asia-Pacific Conference on Communications (2005), pp. 512–515.

[27] J. M. Zain, M. Clarke, Reversible region of non-interest (RONI) watermarking for authentication of DICOM images, International Journal of Computer Science and Network Security 7 (2007) 19–28.

[28] X. Guo, T. ge Zhuang, A region-based lossless watermarking scheme for enhancing security of medical data, J. Digital Imaging 22 (2009) 53–64.

[29] M. Kundu, S. Das, Lossless ROI medical image watermarking technique with enhanced security and high payload embedding, in: 2010 20th International Conference on Pattern Recognition (ICPR), pp. 1457–1460.

[30] E. Cavero, A. Alesanco, L. Castro, J. Montoya, I. Lacambra, J. Garcia, SPIHT-based echocardiogram compression: Clinical evaluation and recommendations of use, IEEE Journal of Biomedical and Health Informatics 17 (2013) 103–112.

[31] J. M. Barton, Method and apparatus for embedding authentication information within digital data, 1997. U.S. Patent 5 646 997.

[32] J. Tian, Reversible data embedding using a difference expansion, IEEE Transactions on Circuits and Systems for Video Technology 13 (2003) 890–896.

[33] C. De Vleeschouwer, J.-F. Delaigle, B. Macq, Circular interpretation of bijective transformations in lossless watermarking for media asset management, IEEE Transactions on Multimedia 5 (2003) 97–105.

[34] H. Trichili, M. Boublel, N. Derbel, L. Kamoun, A new medical image watermarking scheme for a better telediagnosis, in: 2002 IEEE International Conference on Systems, Man and Cybernetics, volume 1, pp. 556–559.

[35] G. Coatrieux, M. Lamard, W. Daccache, W. Puentes, C. Roux, A low distorsion and reversible watermark: application to angiographic images of the retina, in: 27th Annual International Conference on Engineering in Medicine and Biology Society, IEEE-EMBS 2005, pp. 2224–2227.

[36] G. Coatrieux, C. Le Guillou, J.-M. Cauvin, C. Roux, Reversible watermarking for knowledge digest embedding and reliability control in medical images, IEEE Transactions on Information Technology in Biomedicine 13 (2009) 158–165.

[37] C. K. Tan, J. C. Ng, X. Xu, C.-L. Poh, Y. L. Guan, K. Sheah, Security protection of DICOM medical images using dual-layer reversible watermarking with tamper detection capability, Journal of Digital Imaging (2011) 528–540.

[38] X. Guo, T. ge Zhuang, Lossless watermarking for verifying the integrity of medical images with tamper localization, Journal of Digital Imaging 22 (2009) 620–628.

[39] A. Panayides, M. Pattichis, C. Pattichis, C. Loizou, M. Pantziaris, A. Pitsillides, Atherosclerotic plaque ultrasound video encoding, wireless transmission, and quality assessment using h.264, IEEE Transactions on Information Technology in Biomedicine 15 (2011) 387–397.

[40] YCbCr, ITU-R Recommendation BT.601-7: Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios, 2011. Accessed in February 2015, `http://goo.gl/ZL4z5p`.

[41] A. N. Akansu, M. J. Medley (Eds.), Wavelet, Subband, and Block Transforms in Communications and Multimedia, Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[42] A. Said, W. A. Pearlman, A new, fast, and efficient image codec based on set partitioning in hierarchical trees, IEEE Transactions on Circuits and Systems for Video Technology 6 (1996) 243–250.

[43] A. Cohen, I. Daubechies, J. Feauveau, Biorthogonal bases of compactly supported wavelets, Communications on Pure and Applied Mathematics 45 (1992) 485–560.

[44] T. Helleseth, Golomb's randomness postulates, in: Encyclopedia of Cryptography and Security, Springer, 2011, pp. 516–517.

[45] M. Murase, Linear feedback shift register, 1992. US Patent 5,090,035.

[46] L. M. Adleman, H. W. Lenstra, Finding irreducible polynomials over finite fields, in: Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing, STOC '86, ACM, 1986, pp. 350–355.

[47] F. Ono, W. Rucklidge, R. Arps, C. Constantinescu, JBIG2-the ultimate bi-level image coding standard, in: Proceedings of the International Conference on Image Processing (2000), volume 1, pp. 140–143.

[48] Wei Dai, Speed benchmarks for some common cryptographic algorithms, Crypto++ v5.6.0. Accessed in February 2015, `http://goo.gl/XJXbtr`, March 2009.

[49] Certicom Research, Standards for efficient cryptography, SEC 1: Elliptic Curve Cryptography, Version 2.0. Accessed in February 2015, `http://goo.gl/pGo7TP`, May 2009.

[50] D. W. Kravitz, (FIPS PUB 186-2: Digital Signature Standard (DSS). Accessed in February 2015 `http://goo.gl/GgSDKe`, January 2000.

[51] RSA Laboratories, PKCS 1: RSA Cryptography Standard. Accessed in February 2015, `http://goo.gl/13HJ4i`, June 2002.

[52] J. L. Muñoz, J. Forné, J. C. Castro, Evaluation of certificate revocation policies: OCSP vs. Overissued-CRL, in: 2012 23rd International Workshop on Database and Expert Systems Applications, IEEE Computer Society, pp. 511–511.

[53] J. Daemen, V. Rijmen, FIPS PUB 197: Advanced Encryption Standard (AES). Accessed in February 2015, `http://goo.gl/zrVGup`, Nov. 2001.

[54] W. Burr, Selecting the Advanced Encryption Standard, IEEE Security and Privacy 1(2) (2003) 43–52.

[55] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, in: Advances in Cryptology, Springer, pp. 10–18.

[56] T. H. Oh, R. Besar, Medical image compression using JPEG-2000 and JPEG: A comparison study, Journal of Mechanics in Medicine and Biology 2 (2002) 313–328.

[57] DICOM sample image sets, provided by OsiriX DICOM Viewer, 2013. Accessed in February 2015, `http://goo.gl/PyqaMW`.

[58] R. W. Hamming, Error detecting and error correcting codes, Bell System Technical Journal 29 (1950) 147–160.

[59] Z. Wang, A. Bovik, H. Sheikh, E. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Transactions on Image Processing 13 (2004) 600–612.

[60] Z. Wang, Software implementations of the SSIM Index for Image Quality Assessment, 2011. Accessed in February 2015,

`http://goo.gl/VNVTUe`.

[61] R. Bose, D. Ray-Chaudhuri, On a class of error correcting binary group codes, Information and Control 3 (1960) 68–79.

[62] C. E. Shannon, Communication theory of secrecy systems*, Bell system technical journal 28 (1949) 656–715.

[63] T. Kalker, Considerations on watermarking security, in: 2001 IEEE Fourth Workshop on Multimedia Signal Processing, pp. 201–206.

[64] T. H. N. Le, K. H. Nguyen, H. B. Le, Literature survey on image watermarking tools, watermark attacks and benchmarking tools, in: 2010 Second International Conferences on Advances in Multimedia, pp. 67–73.

[65] M. Tanha, S. Torshizi, M. Abdullah, F. Hashim, An overview of attacks against digital watermarking and their respective countermeasures, in: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), pp. 265–270.