J Med Syst (2016) 40: 156
DOI 10.1007/s10916-016-0514-5

Sensitive Patient Data Hiding using a ROI Reversible Steganography Scheme for DICOM Images

Petros L. K. Mantos[1] · **Ilias Maglogiannis**[1]

Received: 4 January 2016 / Accepted: 29 April 2016 / Published online: 11 May 2016

Abstract The exchange of medical images over the Internet has evoked significant interest over the past few years due to the introduction of web and cloud based medical information systems. The protection of sensitive data has always been a key indicator in the performance of such systems. In this context, this work presents an algorithm developed for Digital Imaging and Communications in Medicine (DICOM) medical images, which applies secret-sharing steganography methods for ensuring the integrity of sensitive patient data as well as the important parts of the image. In the proposed algorithm, images are divided into two parts: the region of interest (ROI) and the region of non interest (RONI). Patient data and integrity hashes are positioned inside the ROI while the information (map) needed to recover the ROI before insertion is positioned in the RONI. Security of the extraction process is assured through the use of cryptography. The experimental results prove that the original (cover) images and the stego images provide an excellent visual equality result in terms of PSNR. Furthermore, they prove that the proposed scheme can be efficiently used as a steganography scheme in DICOM images with limited smooth areas.

Keywords Watermarking · Steganography · Secret sharing · DICOM medical images · Security · Image processing

This article is part of the Topical Collection on *Transactional Processing Systems*

✉ Ilias Maglogiannis
imaglo@unipi.gr

[1] Department of Digital Systems, University of Piraeus, Grigoriou Lampraki 126, PC 18532 Piraeus, Greece

Introduction and related work

Recently Web and Cloud based medical information systems have dynamically entered the market of electronic healthcare systems. Thus, the exchange of medical data over the Internet and the storage of data in Cloud infrastructures is prospectively a common practice in modern hospital information systems [1, 2]. In addition, online collaboration systems such as the one presented in [3] require the exchange of medical images over the Internet. A key performance indicator for the adoption of such systems is data security [4]. It must be ensured that medical data is protected both during transmission and at the storage site. In the case of medical images, sensitive data is embedded in image files as header information defined in the Digital Imaging and Communications in Medicine (DICOM) standard [5]. Medical images exchange over the Internet is also specified by the Web Access to DICOM Objects (WADO) standard, which involves a Web-based service for accessing and presenting DICOM persistent objects [5]. More specifically, WADO provides a simple mechanism for accessing a DICOM image from HTML pages or XML documents, through HTTP/HTTPS. Data can be retrieved either in a presentation-ready form as specified by the requestor (e.g., JPEG or GIF) or in a native DICOM format.

One of the main security concerns in the aforementioned case is guaranteeing the integrity and identity of the medical images [6]. Thus, authenticating both the origin of the image as well as proving that it was not modified before and during transmission is of crucial importance. Data hiding (specifically watermarking) techniques exist that may satisfy one or more of the above security requirements [6–9]. These techniques insert additional data inside the corresponding files and alter the images. Thus, an important requirement for exploiting data hiding methodologies is that the image must be of a sufficient size to conceal data. In addition, reversibility

of at least the regions of interest must be applicable, to prevent the loss of any diagnostic information.

As far as image watermarking techniques are concerned, they can be evaluated by the following metrics:

- Capacity: Defined as the amount of information that can be embedded. For example when using simple Least significant bit (LSB) modification the maximum amount of data that can be hidden (in bits), in an 8-bit image is equal to the number of bytes that compose the image.
- Robustness: Defined as the resistance of the embedded information to image changes. For example, when using simple LSB modification, compression of the image would probably change the LSB's of the pixels and thus destroy the data inserted.
- Imperceptibility: Defined as how well the data is hidden with regard to human senses.
- Privacy/Security: Defined as the amount of secret knowledge needed for accessing the embedded information. For example, a secret key may be needed to extract the hidden data [10, 11].

A categorization of watermarking techniques applied in medical images was introduced by Coatrieux et al. in [10]. Three categories of watermarking algorithms are the most prominent. The first class of algorithms embeds the information into the RONI, which is specified as the black background and possibly some grey regions of minor interest. The second class corresponds to reversible watermarking, which allows retrieval of the original image once the watermark is read. Finally, the third class corresponds to classical watermarking techniques which modify small visual aspects like the LSB's of the pixel values, minimizing image distortion.

An additional categorization based on the purpose of medical image watermarking is presented by Al-Qershi et al. in [6]. More specifically, medical image watermarking can be divided into three schemes. The first, called an authentication scheme aims at authenticating the origin of the image while also providing tamper detection and recovery of the pixels that have been altered. The second called data-hiding scheme aims to conceal patient data with high perceptibility. Finally, the third category is a combination of the first two.

As far as steganography is concerned, the main goal is to conceal the presence of hidden communication. Thus, perceptual and algorithmic undetectability is crucial. Furthermore, in image steganography the original image is referred to as the cover image while the image containing the hidden data as the stego image. In addition, with the exception of one major difference, steganography requirements are the same as the watermarking requirements described above. More specifically, in steganography the imperceptibility requirement refers to how well the data is hidden regarding not only visual recognition but also steganalysis algorithms. Nevertheless, it must be noted that while steganography has high demands in capacity and imperceptibility and low demands in robustness, watermarking has high demands in robustness and low demands regarding capacity [11].

One of the most common steganography methods is LSB replacement, which is also known as LSB substitution. This method simply overwrites the LSB's of the pixel values with the bit of insertion while the pixels to be manipulated are randomly chosen using a Pseudorandom number generator (PRNG). Thus, the manipulated LSB of the pixel value will either be modified or remain unaltered. However, by embedding a uniformly distributed message, statistical patterns may appear on the image's histogram leading to easy detection [12]. Another technique, named LSB matching, exists which overcomes this issue by randomly applying a +1 or −1 operation when a pixel value must be modified. Furthermore, a method known as optimal pixel adjustment process (OPAP) was introduced by Chan et al. in [13] which provided much improved stego image quality compared to LSB replacement. While the above methods use only one pixel as an embedding unit, methods known as pixel pair matching (PPM) methods which use pairs of pixels to hide the message also exist. The message in these methods is represented using a specific B-ary notational system. A clear example is LSB matching revisited (LSBMR) introduced by Mielikainen in [12], which considers pairs of pixels and their relationship for insertion. More specifically, the LSB of the first pixel carries one bit while the odd-even relationship between the pixels is used to extract the other. LSBMR provides a mean square error (MSE) of 0.375 for an embedding rate of 1 bpp, which is a significant advantage compared to the MSE of 0.5 which LSB replacement provides [12]. In addition, improvements to LSBMR were introduced such as the exploiting modification direction (EMD) method formed by Zhang et al. in [14]. In EMD only one pixel in a pixel pair is changed, one gray-scale unit maximum, and a message digit in a 5-ary notational system can be embedded. Thus, a maximum payload of $(1/2)*\log_2 5 \approx 1.161$ bpp is provided [14]. Furthermore, the diamond encoding (DE) method, which enhanced the payload of EMD, was introduced by Chao et al. in [15]. In addition, the adaptive pixel pair matching (APPM) formed by Hong et al. in [16] provided an even lower distortion rate than DE. The above algorithms do not take into account the way human sight reacts to changes in pixels with diverse content, such as smooth blocks of pixels and edges. Nevertheless, edge adaptive algorithms exist. One of the most common schemes used in such algorithms is pixel value differencing (PVD) in which the number of bits to be embedded is calculated based on the difference between a pixel and its neighbors [17–21]. Greater

differences in relation to one's neighbor indicate that more bits can be embedded [17]. A recently published method for steganography is the "Dual-Level Security based Cyclic18 Steganographic" method presented by Muhammad et al. in [22]. This method applies a variety of methods such as image scrambling, the use of custom encryption schemes and the insertion of the encrypted message using LSB and intermediate LSB substitution.

The algorithm proposed in this work is an edge adaptive scheme, which also belongs to the category of Secret Sharing algorithms. It is utilized using the Secret Sharing methods given by Yuan in [23]. A (n,k) Secret Sharing scheme/algorithm "inserts" the message into k shares. Subsequently, extraction of the message can be accomplished by using $x \geq k$ shares, while extraction using less than k shares is computationally impossible. Furthermore, none of the k shares themselves reveal any information about the message. The proposed algorithm is applied on DICOM images and inserts sensitive patient data, recovery data needed to recover the ROI part of the image and authentication data needed to validate the integrity of the ROI of image and the patient's data. The proposed implementation takes into account the fact that DICOM images are of considerable size and contain large RONI with zero diagnostic value.

The rest of the paper is structured as follows: In Section 2 we present an overview and the distinct modules of the insertion and extraction schemes, while in Section 3 use case application of the proposed insertion scheme in practice is provided. In Section 4 experimental results are provided and in Section 5 we discuss the results and conclude the paper.

The proposed steganography methodology

In this Section we provide the technical details of the proposed steganography method for the DICOM medical images. In the rest of this paper the following conventions are made:

- Sensitive patient data (extracted from the DICOM Tags included in image header) that is chosen for insertion is referred to as *'data'*.
- Data used to recover the ROI part of the image is referred to as *'recovery data'*. More specifically the *'recovery data'* consists of:

  – The *'map'* which is a mapping of the pixels modified in the ROI during insertion and is crucial for the recovery of the ROI.
  – The *'size of map'* which is a variable that represents the size of the *'map'*. This variable consists of 16 bits.
  – The *'start row'* which contains the value of the starting row of the ROI. This variable consists of 10 bits. For example if this variable has the value '3' it is represented as '0000000101'. It must be noted, that 16 and 10 bits are more than adequate to store the values of the *'size of map'* and *'start row'* respectively.

- Data used to validate the origin and the integrity of the image is referred to as *'authentication data'*. The *'authentication data'* actually consists of two hashes, one for the ROI part of the image (*'hash ROI'*) and one for the *'data'* hidden (*'hash data'*).
- The encrypted form of the *'data'* and *'authentication data'* is referred to as *'encrypted data'*. More specifically, the above are encrypted to ensure confidentiality even if steganography is detected.

The insertion scheme

An overview of the modules that comprise the proposed insertion scheme is illustrated in Fig. 1.

The following steps comprise the proposed insertion algorithm:

Insertion of the original image

At this step the user inputs the medical image in DICOM format. The data to be hidden consists of the sensitive elements of the DICOM header (i.e. patient name and demographic data, referring physician data etc.). The exact definition of data to be hidden is achieved in the next step either manually by the user or using pre-defined rules.

DICOM header fields selection

In this step the fields of the DICOM header which need to be hidden and anonymized are determined. The majority of the DICOM header sensitive information is placed under the patient information group ('0010'). Nevertheless, additional fields exist which could be used in order to identify the patient directly or indirectly through a combination of other fields. The supplement 142 of the DICOM standard provides guidelines that must be followed in order to properly achieve de-identification of an image. Because, the list of fields that must be anonymized is immense (as provided in the supplement 142 of the DICOM standard) lists such as the one provided in [24] can be used. This list (see Table 1) was utilized in order to evaluate the efficiency of 10 free DICOM anonymizers and thus adequately describes a minimum subset of fields that should be anonymized in order to protect the identity of the patient. As is obvious, there is no need to store both the group-

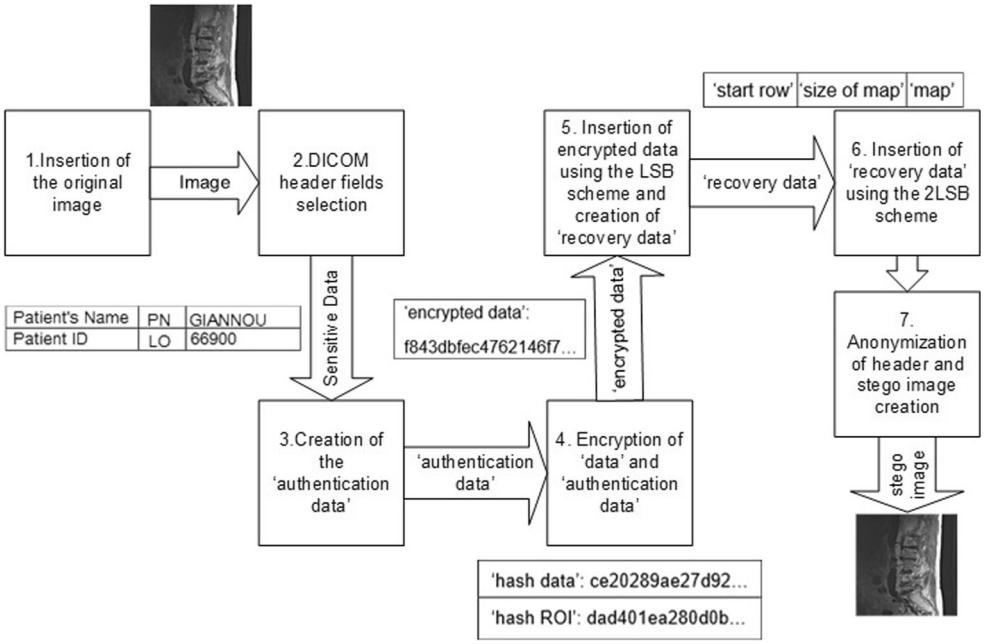


Fig. 1 Modules that compromise the insertion scheme

element and Tag description values, as only one of them is required to define the field.

After the selection of fields, the algorithm is able to determine the maximum available ROI. This is because it has been defined in this paper as the part of the image that does not belong to the RONI which can subsequently be calculated when the '*data*' is selected. Nevertheless, how the RONI size is defined will be clarified below. In this scheme, the maximum ROI is considered; however an even smaller ROI could be specified manually. At Fig. 2 an example of the ROI and RONI is given. More specifically the RONI contains two parts: one that consists of the first rows of the image which will be referred to as 'RONI part 1' in the rest of this paper and one of the last rows which will be referred to as 'RONI part 2'. The above describes the way in which the ROI and RONI are defined throughout this paper. Nevertheless, if the first or last rows of the image contain valuable information, different regions (and subsequently shares) can be selected with the sole restriction that information needed in order to locate every share must be provided to the user (i.e windows at the corners). As aforementioned, in our case this information is provided through the "start row" variable. This variable could be substituted to include points (vertexes), which will point to the shares of the ROI and RONI.

Creation of the 'authentication data'

In the proposed scheme, calculation of the '*authentication data*' must be accomplished using a secure cryptographic hash function due to the significance of the data hidden. Nevertheless, considering the fact that the main goal of this algorithm is to ensure integrity and since hashes are encrypted and thus only retrievable by the legitimate user, there is no need to choose a secure cryptographic hash function, which although more secure is less efficient. In the proposed scheme two hash values '*hash data*' and '*hash ROI*' are calculated in order to confirm the integrity of the patient's data and of the ROI part of the image respectively.

Encryption of 'data' and 'authentication data'

The encryption of the '*data*' and '*authentication data*' can be implemented using any private or public key cryptography algorithm. Obviously, the user must have inserted the equivalent (encryption) key prior to this step. The choice of public key cryptography could be used in a scenario in which the equivalent keys have been assigned to doctors of a specific hospital. Thus, by applying encryption with the public key of the recipient it would be assured that only that specific user (using his private key) could decrypt the data. The adverse aspect of this is the high demand of resources regarding public key cryptography in comparison to private key cryptography. Furthermore, the choice of private key cryptography could be used in a scenario in which the key is also used as a session key during transmission of the image among the communicating parts.

A description of how the maximum available ROI is calculated is presented below. It should be noted that the '*recovery data*' is inserted at the RONI, while the '*encrypted data*' at the ROI.

Supposing that the '*data*' is x bits of size and every hash consists of y bits, there will be a total of $x+2*y$ unencrypted data bits. Furthermore, if a stream cipher is used the encrypted data will also occupy $x+2*y=z$ bits. In contrast, if a block

Table 1 Minimum subset of fields that should be anonymized in order to protect the identity of the patients

Tag ID	Tag Name
0008,0020	StudyDate
0008,0021	SeriesDate
0008,0022	AcquisitionDate
0008,0023	ContentDate
0008,0024	OverlayDate
0008,0025	CurveDate
0008,002A	AcquisitionDatetime
0008,0030	StudyTime
0008,0031	SeriesTime
0008,0032	AcquisitionTime
0008,0033	ContentTime
0008,0034	OverlayTime
0008,0035	CurveTime
0008,0050	AccessionNumber
0008,0080	InstitutionName
0008,0081	InstitutionAddress
0008,0090	ReferringPhysiciansName
0008,0092	ReferringPhysiciansAddress
0008,0094	ReferringPhysiciansTelephoneNumber
0008,0096	ReferringPhysicianIDSequence
0008,1040	InstitutionalDepartmentName
0008,1048	PhysicianOfRecord
0008,1049	PhysicianOfRecordIDSequence
0008,1050	PerformingPhysiciansName
0008,1052	PerformingPhysicianIDSequence
0008,1060	NameOfPhysicianReadingStudy
0008,1062	PhysicianReadingStudyIDSequence
0008,1070	OperatorsName
0010,0010	PatientsName
0010,0020	PatientID
0010,0021	IssuerOfPatientID
0010,0030	PatientsBirthDate
0010,0032	PatientsBirthTime
0010,0040	PatientsSex
0010,1000	OtherPatientIDs
0010,1001	OtherPatientNames
0010,1005	PatientsBirthName
0010,1010	PatientsAge
0010,1040	PatientsAddress
0010,1060	PatientsMothersBirthName
0010,2150	CountryOfResidence
0010,2152	RegionOfResidence
0010,2154	PatientsTelephoneNumbers
0020,0010	StudyID
0038,0300	CurrentPatientLocation
0038,0400	PatientsInstitutionResidence
0040,A120	DateTime
0040,A121	Date
0040,A122	Time
0040,A123	PersonName

Source: [24]

cipher is used the encrypted data will occupy $\lceil (x+2{*}y)/block\_size \rceil {*} block\_size = z$ bits. In addition, taking into account that 2 bits per map entry is needed and that 10 bits are used to represent the '*start row*' and 16 bits to represent the '*size of map*' there will be a total of 2*z+26 bits of insertion at the RONI. Furthermore, insertion of the '*start row*' takes place considering two shares (share 1 and share 2) by dividing the 'first part of the RONI' vertically and the insertion of the '*size of map*' and '*map*' takes place considering four shares (shares 1 to 4) by dividing both parts of the RONI vertically (see Fig. 3). Thus, since using the 2LSB scheme two bits per pixel are inserted, the RONI consists of $\lceil (z+13)/\lfloor (\frac{width\_of\_image}{2}) \rfloor \rceil$*2 rows and the ROI consists of the remaining image rows. The reason why the '*start row*' is inserted using only the first two shares of the RONI is that there is no possible way of knowing where the remaining shares (3 and 4) start as this information is given by the '*start row*' variable itself. To be more precise, ROI starts at 'start row' and ends at 'start row' rows before the last row of the image. Thus, if the image consists of rows 0 to *N*-1, (where *N* is equal to the rows in the image), then shares 1 and 2 of the RONI end at ('*start row*' −1) row while shares 3 and 4 of the RONI start after the last row of the ROI.

Insertion of 'encrypted data' using the LSB scheme and creation of 'recovery Data'

The insertion of the '*encrypted data*' is done adopting the LSB method presented by Yuan in [23]. Two methods of insertion are proposed, the former with the ability to manipulate the LSB's of the cover (original) images while the latter has the ability to manipulate both the LSB's and second LSB's. The Author proved that these methods had excellent results compared to steganalysis when natural images are used. In this work, slightly modified versions of these methods are applied to medical images. As stated in [23], the complexity of both the LSB and 2LSB (n,n) secret sharing methods are proportional to the amount of data hidden and the number of shares used. Thus, for both methods, if the amount of data inserted is equal to "*x*" and "*n*" number of shares are used, the computational complexity of each method is O(*x***n*). The same applies to the slightly modified versions used.

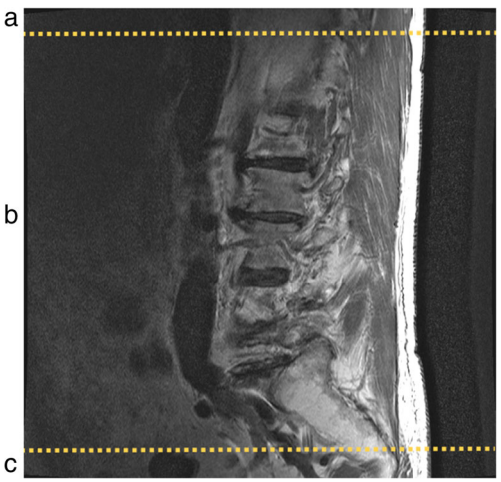


Fig. 2 Example of ROI and RONI definition: (**a**) RONI part 1, (**b**) ROI, (**c**) RONI part 2

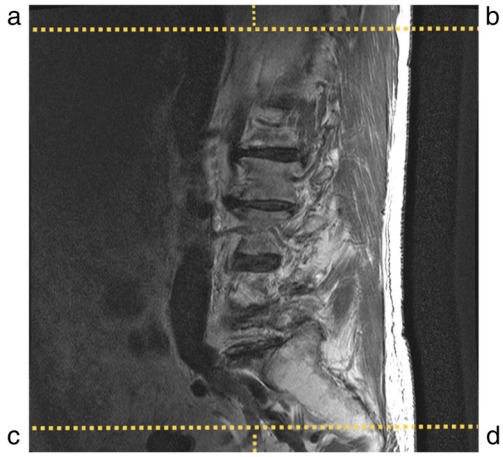


Fig. 3 Available shares in the RONI: (**a**) share 1, (**b**) share 2, (**c**) share 3 and (**d**) share 4

The LSB method first creates the shares that will be used, for example the pixels that are comprised in every cover image. Then the Sobel operators for each share are calculated [25]. Afterwards, insertion of the message occurs bit by bit. More specifically, for every bit of insertion the XOR values of the corresponding LSB's of the pixels in every share is calculated. If the result differs from the current bit of insertion then LSB matching is applied to the pixel which has the largest gradient magnitude among the pixels of the shares that are currently processed. This guarantees that the pixel being more robust against steganalysis will be modified. Reconstruction of the hidden message is simply achieved by applying XOR operations in the LSB's of the pixels of the stego shares [23].

In order to reconstruct the original ROI, the proposed scheme requires a way of knowing if a +1 or −1 operation was applied, since LSB matching does this in a random manner. To address this issue a PRNG binary is created using a secure PRNG and the cryptographic key provided by the user as a seed. A PRNG produces a sequence of pseudo numbers. Furthermore, by using the same seed the same binary sequence will be produced during insertion and extraction. When a pixel needs to be modified the equivalent value of the PRNG binary is checked. If this value is '1', then a '+1' operation is applied otherwise a '−1' operation. At this point it must be noted that, if public key cryptography is used, using only the public key of the recipient as a seed in the PRNG, will produce the same PRNG binary values every time. To address this issue, a value could be added to the seed, or could be used alone instead of the public key. This value should be inserted with the rest of the data so it is available during extraction.

The encrypted data is inserted using the LSB method and the PRNG binary starting at '*start row*' which defines the starting row of the ROI. Two shares are considered by dividing the ROI vertically. This step will produce the extraction '*map*' needed for the reconstruction of the image and will contain 2 bits for every bit inserted.

More specifically the combination:

- '00', states that no modification occurred.
- '01', states that a modification occurred in the first half (share 1) of the ROI.
- '11', states that a modification occurred in the second half (share 2) of the ROI.
- '10', is used to state that a modification of a border value took place using the opposite value to the one stated from the PRNG binary. For example if the PRNG binary states that a −1 operation must take place and the value of the chosen pixel is 0 then a +1 will actually be held, to avoid an underflow. Thus, during reconstruction, information is needed to apply a −1 operation and not a +1. During reconstruction, the reconstruction process will check the pixel values of both shares and will apply a −1 operation to the share that has the value '1'. At this point it must be noted that a special condition occurs if the pixel value of the second share has the value '1', since, during reconstruction it would be

Table 2 Description of pixel manipulation and map values during LSB insertion based on the values of the PRNG binary

Value and insertion action pointed by the PRNG binary	Value of first Share before insertion	Value of second Share before insertion	Action applied	Value of first Share after insertion	Value of second Share after insertion	Map value
0→−1 operation	$x \neq 0$	$y \in [0,255]$	$x = x-1$	$x-1$	y	01
0→−1 operation	$x = 0$	$y \neq 1$	$x = x+1$	$x = 1$	y	10
0→−1 operation	$x = 0$	$y = 1$	$y = y-1$	$x = 0$	$y = 0$	11
1→+1 operation	$x \neq 255$	$y \in [0,255]$	$x = x+1$	$x+1$	y	01
1→+1 operation	$x = 255$	$y \neq 254$	$x = x-1$	$x = 254$	y	10
1→+1 operation	$x = 255$	$y = 254$	$y = y+1$	$x = 255$	$y = 255$	11

It is assumed that the first share has the biggest gradient magnitude among the pixels chosen. Grayscale pixel values of 8 bits are considered

a	b	c
0000101000	0000001101010101	1101001000…

Fig. 4 Example of data for insertion using the 2LSB scheme: (**a**) 'start row', (**b**) 'size of map', (**c**) 'map'

impossible to ascertain which pixel should be modified. Thus, in this rather rare case a '−1' operation is held at the pixel of the share that was not chosen for manipulation and the map value is subsequently set to '01' or '11'. The equivalent applies in the case of possible overflow.

Table 2 displays the operations that take place for 8-bit grayscale images during the LSB insertion and the equivalent map values. In the table it is assumed that the first share has the biggest gradient magnitude among the chosen pixels. Nevertheless, the equivalent operations take place when the pixels of the second share have the biggest gradient magnitude among the pixels chosen for manipulation.

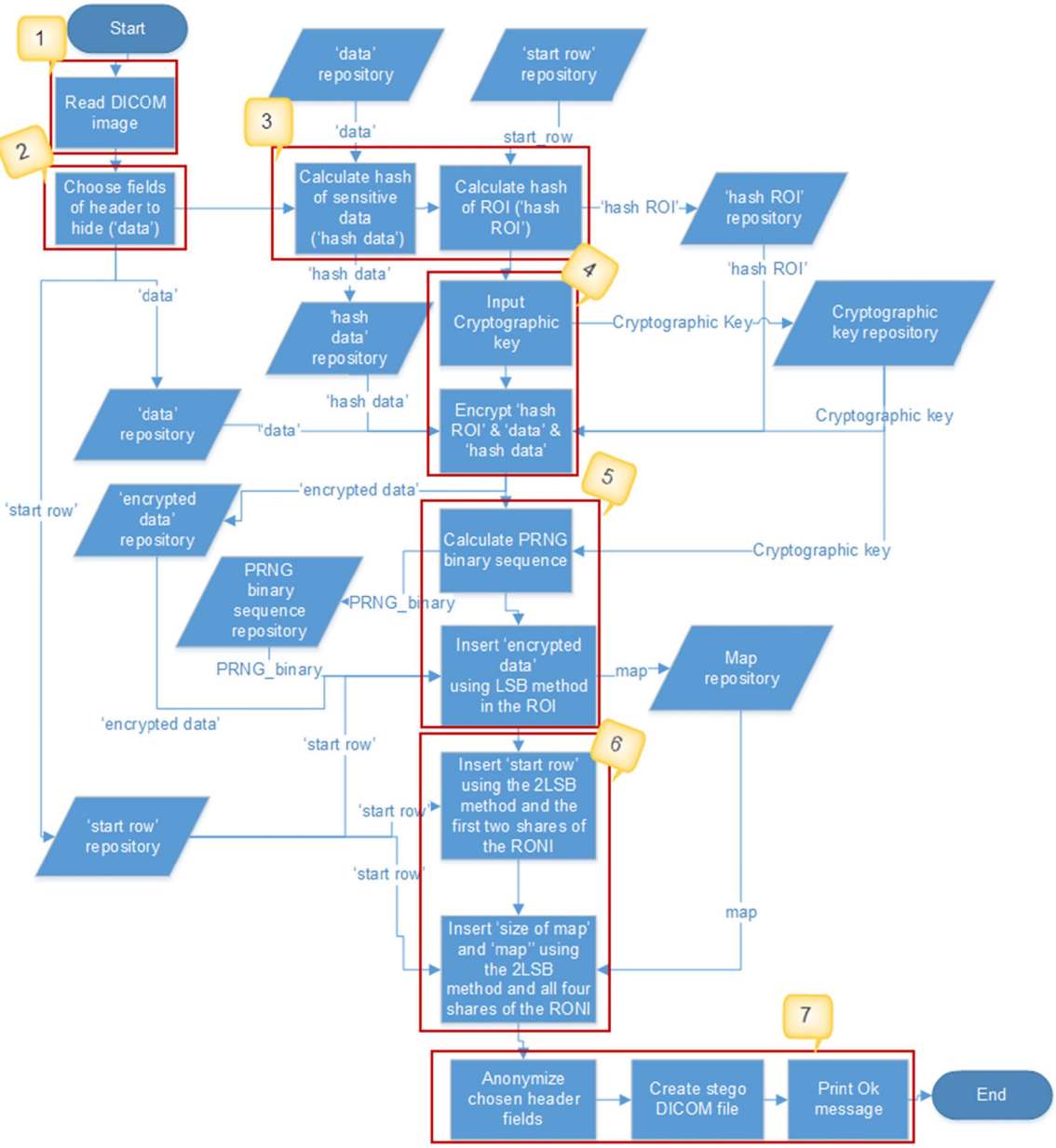


Fig. 5 Workflow of the insertion scheme: (**1**) Insertion of the original image, (**2**) DICOM header tags selection, (**3**) Creation of the 'authentication data', (**4**) Encryption of 'data' and 'authentication data', (**5**) Insertion of 'encrypted data' using the LSB scheme and creation of 'recovery data', (**6**) Insertion of 'recovery data' using the 2LSB scheme, (**7**) Anonymization of header and stego image creation

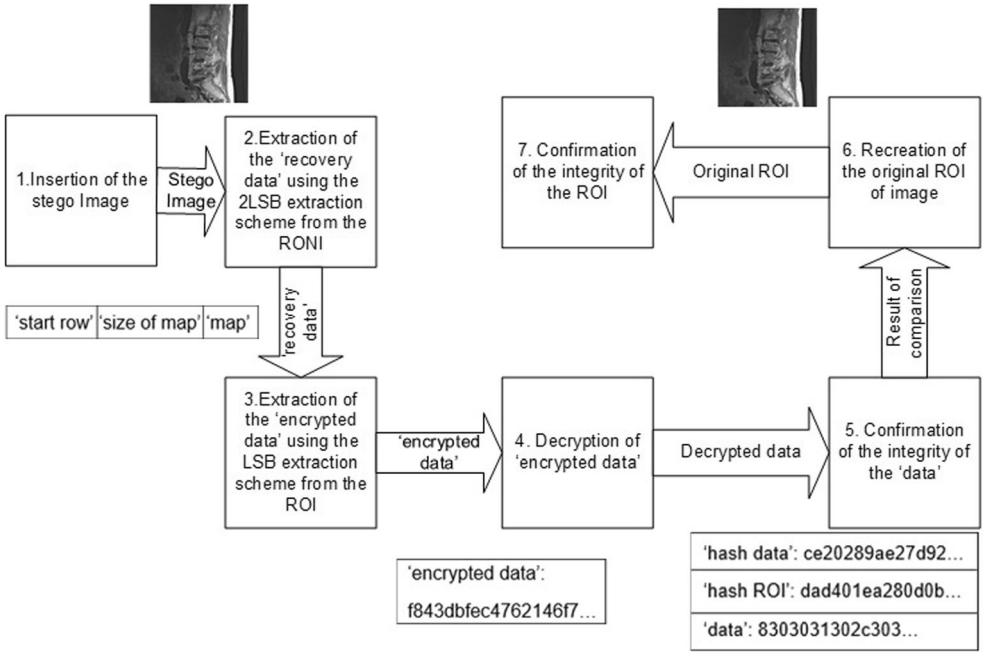


Fig. 6 Modules that compromise the extraction scheme

Insertion of 'recovery data' using the 2LSB scheme

As stated earlier, Yuan in [23] also proposed a method which can modify both the LSB and the second LSB of the pixel values. In this step the '*recovery data*' is inserted using this 2LSB insertion method. More specifically, the '*recovery data*' consists of (see Fig. 4):

- The ROI starting row ('*start row*') so it is known at which row extraction of the data using the LSB extraction method should start. This component is inserted using only the two shares that belong to the first part of the RONI.
- The maps size ('*size of map*') so during extraction using the 2LSB extraction method the ending point of the extraction is known. This component is inserted using all shares of the RONI.
- The '*map*' so that the ROI part of the image can be reconstructed. This component is inserted using all shares of the RONI.

Anonymization of the header and stego image creation

At this step, the fields that the user chose for insertion are anonymized. The detailed workflow of the insertion scheme is presented in Fig. 5.

Table 3 Description of pixel manipulation during recovery of the ROI, based on the PRNG binary and extraction map values

Value and extraction action pointed by the PRNG binary	Map value	Value of first Share before insertion	Value of second Share before insertion	Action applied	Value of first Share after insertion	Value of second Share after insertion
0→ +1operation	01	x	y	$x=x+1$	$x+1$	y
0→ +1operation	11	x	y	$y=y+1$	x	$y+1$
0→ +1operation	10	$x\neq 1$	$y=1$	$y=y-1$	x	$y=0$
0→ +1operation	10	$x=1$	$y\neq 1$	$x=x-1$	$x=0$	y
1→ −1operation	01	x	y	$x=x-1$	$x-1$	y
1→ −1operation	11	x	y	$y=y-1$	x	$y-1$
1→ −1operation	10	$x\neq 254$	$y=254$	$y=y+1$	x	$y=255$
1→ −1operation	10	$x=254$	$y\neq 254$	$x=x+1$	$x=255$	y

Grayscale pixels values of 8 bits are considered

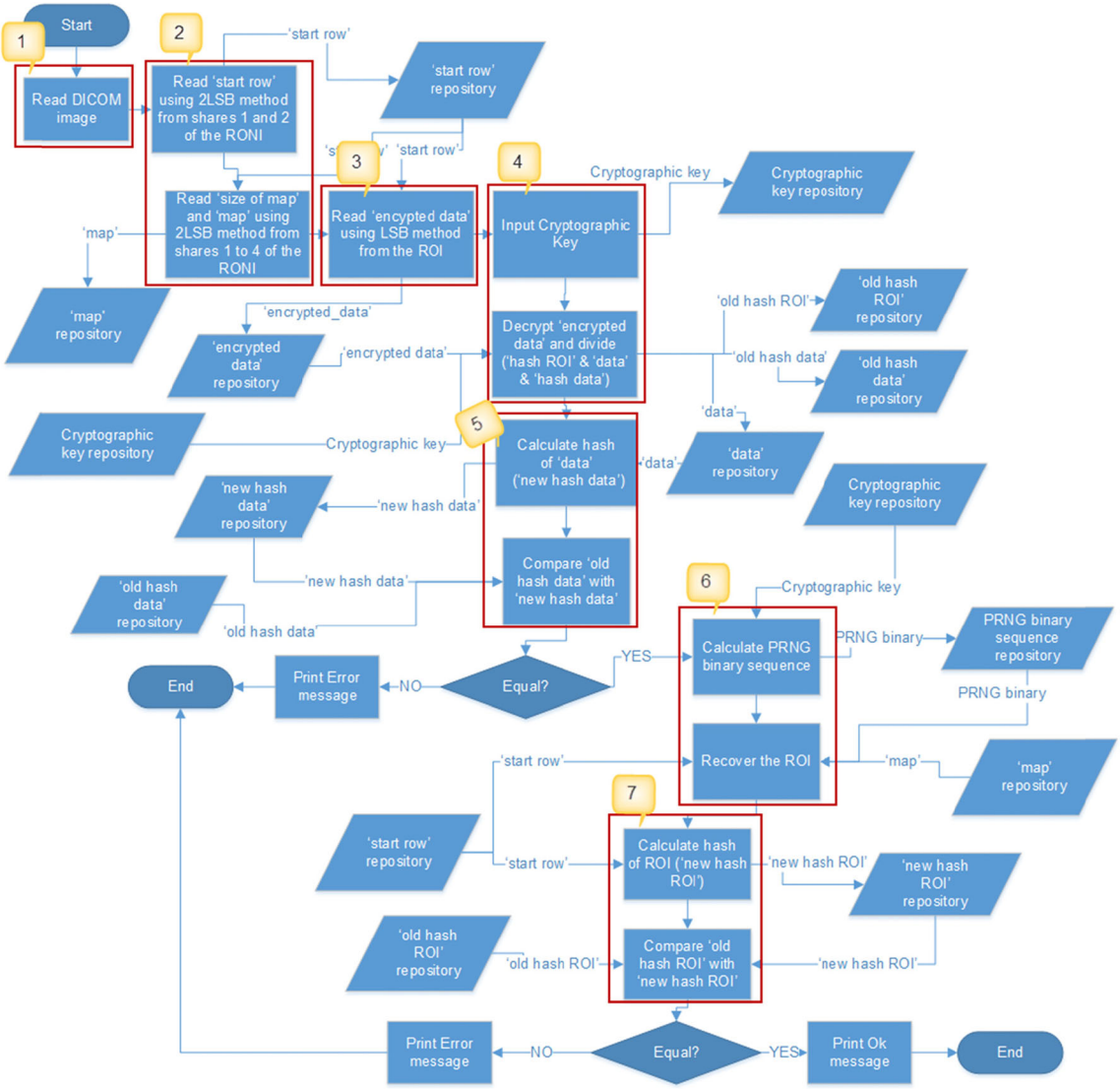


Fig. 7 Workflow of the extraction scheme: (**1**) Insertion of the stego image, (**2**) Extraction of the 'recovery data' using the 2LSB extraction scheme from the RONI, (**3**) Extraction of the 'encrypted data' using the LSB extraction scheme from the ROI, (**4**) Decryption of 'encrypted data', (**5**) Confirmation of the integrity of the 'data', (**6**) Recreation of the original ROI of image, (**7**) Confirmation of the integrity of the ROI

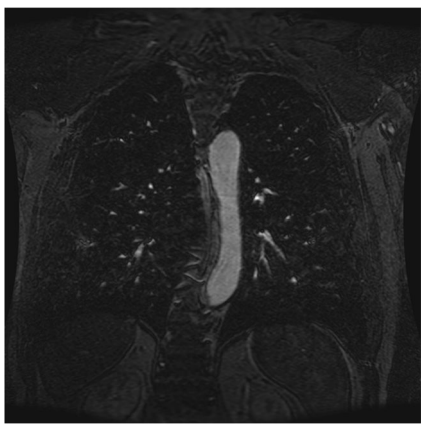


Fig. 8 Medical Image in DICOM format tested at the use case

The extraction scheme

In this section an overview of the modules that comprise the proposed extraction scheme is presented. More specifically the extraction scheme can be divided into the following modules illustrated in Fig. 6:

Insertion of stego image

At this step the stego medical image is inserted by the user.

Extraction of the 'recovery data' using the 2LSB extraction scheme from the RONI

At this step starting at the first row, extraction of the '*recovery data*' using the 2LSB extraction method takes

Table 4 Sensitive patient data of header, chosen for anonymization at the use case

Group	Element	Tag description	VR	Length	Value
0010	0010	Patient's Name	PN	12	Papadopoulos
0010	0020	Patient ID	LO	6	66900
0010	0030	Patient's Birth Date	DA	8	20080822
0010	0040	Patient's Sex	CS	2	M
0010	1000	Other Patient IDs	LO	6	66900
0010	1030	Patient's Weight	DS	4	130
0010	1040	Patient's Address	LO	10	Pagalou 29
0010	1060	Patient's Mother's Birth Name	PN	8	Katerina
0010	2150	Country of Residence	LO	6	Greece
0010	21F0	Patient's Religious Preference	LO	8	Catholic

The data does not correspond to a real patient

place. More specifically, the following components are extracted:

- The first row of the ROI ('*start row*'). So that the starting row from where extraction of the encrypted data will take place, as well as the rows that constitute the 'RONI part 2', are known. This component is extracted using only the two shares that belong to the first part of the RONI.
- The size of the map ('*size of map*'). So that the algorithm knows when to stop extracting the map and subsequently the encrypted data. This component is extracted using all of the shares of the RONI.
- The '*map*', so that the ROI can be recovered. This component is extracted using all of the shares of the RONI.

Extraction of the 'encrypted data' using the LSB extraction scheme from the ROI

At this step, starting at the '*start row*' row and for '*size of map*' bits, extraction of the encrypted data using the LSB extraction method takes place.

Decryption of 'encrypted data'

At this step decryption of the '*encrypted data*' takes place. Again, as is obvious the cryptographic key must have been inserted by the user prior to this step. After the decryption of the '*encrypted data*', the '*data*', '*hash data*' referred to as '*old hash data*' and '*hash ROI*' referred to as '*old hash ROI*' are parted.

Confirmation of the integrity of the 'data'

At this step the hash of the extracted '*data*' (referred to as '*new hash data*') is calculated and is compared with the retrieved hash of the '*data*' ('*old hash data*'). If those values are divergent it means that the integrity of the '*data*' cannot be validated as it has been altered and consequently the user must be informed. Else, if those values are identical then, consequently the integrity of the '*data*' has been successfully validated.

Recreation of the original ROI of image

At this step recalculation of the original ROI of the image occurs. First, the PRNG binary is created using the cryptographic key as a seed. Then using the PRNG binary and the '*map*' the following is applied for every entry of the map. If the map's value is not '00' then a modification during insertion has occurred. Thus, reversal of the operation that took place during insertion is required. Table 3 includes the operations that take place for 8-bit grayscale images during recovery of the ROI.

Confirmation of the integrity of the ROI

At this step the hash of the ROI (referred to as '*new hash ROI*') is calculated and is compared with the retrieved hash of the ROI ('*old hash ROI*'). If those values are divergent it means that the integrity of the ROI cannot be validated as it has been altered and consequently, the user must be informed. Else, if those values are identical the integrity of the ROI has been successfully validated. The overall workflow of the extraction scheme is depicted in Fig. 7

'hash data': d7ec6f25f3176ec2e70c02a4cfc1224a83dd09d28804ff914ba567b30d1a56a5

'hash ROI': 75915ef2846c6f0480b14757007d0c2c643edd76868e0ac8f4f04e035892d6e8

Fig. 9 Authentication data (*'hash data'* and *'hash ROI'*) produced at the use case

Encryption key: 3300e014e929aca1c2e515ae813d55da

Fig. 10 Encryption key used at the use case

The proposed algorithm in practice

In this section, the application of the proposed insertion scheme to DICOM medical images is presented. The DICOM medical image, which will be used as an example, is displayed in Fig. 8.

The DICOM tags/fields selected in this example for anonymization are provided in Table 4. As aforementioned, there is no need to store both the group-element and tag description values. Thus, only the group-element of each field along with its VR and value are stored.

The hash function chosen for the specific implementation is SHA-256, which produces an output of 256 bits. The hash values (represented in hexadecimal) of the sensitive '*data*' and the ROI are given in Fig. 9. The chosen '*data*' size is 226 bytes. Thus, by adding 32 bytes of the '*hash data*' and 32 bytes of the '*hash ROI*' a total of 290 bytes or otherwise 2320 bits of unencrypted data occurs. Furthermore, as the application was developed in JAVA the JAVA Secure random generator (SecureRandom class) was used to produce the PRNG binary.

In the encryption step of the '*data*' and '*authentication data*' AES-128 is used, which is a block cipher that uses a block size of 128 bits [26]. The key used (represented in hexadecimal) is given in Fig. 10. In addition, the '*encrypted data*' is given in Fig. 11. The rationale behind the choice of AES as the encryption algorithm relies on the fact that it is assumed that the key is used as a session key during transmission of the image. It must be noted that algorithms like blowfish are faster than AES [27]. On the other hand, given that the data that is encrypted is of a small size and that AES is a standard, it is considered to be the optimum choice. The size of the '*encrypted data*', which as mentioned is used to calculate the '*size of map*' and subsequently the maximum ROI is calculated as follows:

Since the unencrypted data occupies 2320 bits, $\lceil 2320/128 \rceil = 19$ blocks or otherwise $19*128 = 2432$ bits of '*encrypted data*' is produced. Furthermore, since the image used occupies 512 rows and 512 columns by dividing the image in half vertically a maximum insertion of $512/2 = 256$ bits at every row using the LSB method and $256*2 = 512$ bits using the 2LSB method occurs. Thus, $\lceil (2432+13)/256 \rceil * 2 = 20$ rows are needed for the insertion of the '*recovery data*'. Furthermore, it has been chosen not to take into account the first and last row of pixels due to the fact that they have a gradient magnitude of zero. Thus, the first part of the RONI will consist of the rows 1 to 10 and the second part of the RONI will consist of the rows 501–510. Thus, the insertion of the '*encrypted data*' and consequently the maximum ROI must start at the 12th row (row 11) and end at the 501th row (row 500). Thus, the '*start row*' value is 11.

In the specific example, data insertion using the LSB method begins at the 12th row. In Fig. 12, the values produced by the Sobel operators corresponding to the first 60 rows of the image and the pixels that were chosen for manipulation using the LSB method (represented with white) are illustrated. It is obvious that the left and right part of the image were not manipulated because they consist of smooth areas. In contrast, manipulation took place in the central part of the image where the pixels had increased gradient magnitude values. As aforementioned this step produces the '*map*' given at Fig. 13.

The next step involves the insertion of the '*map*' along with the '*size of map*' and the '*start row*' (Fig. 14) at the RONI using the 2LSB method. As aforementioned, the '*start row*' is inserted using the first two shares of the RONI while the '*size of map*' and '*map*' using all four shares. In Fig. 15, the values produced by the Sobel operators corresponding to the first 60 rows of the image and the pixels that were chosen for manipulation using the 2LSB method (represented with white) are depicted. Furthermore, the equivalent values corresponding to the last 60 rows of the image are given in Fig. 16. Once again, it is obvious that the left and right part of the images were not manipulated because they consist of smooth areas. In contrast, manipulation took place in the central part of the image where the pixels had augmented gradient magnitude values.

Finally, the stego image is created.

Fig. 11 'encrypted data' produced by the insertion algorithm at the use case

'encrypted data':
e0c9baeec73850aa53bc145e21bb007ad1065ee3b45131a40a67c14c19071e8
2bc18c75f1ad44eb61121279e1a2efc3ac804efe8b2cb09c3fce1f1459877c97e1
81dc540314996c7513a65ba9c6448c2424784f18bb82fdec636f1f85698b9c40bd
1ddf213f2c4d6faf203bed5d07724b0d66c5d08f821f4204c35f0e7302104d73e50
4e47d34f34e56178974bb5dbfce5ce505df39192f65959439edf411639611651aa
0a86f32a209a96e42c2d09a5e18218795d37b1ee7a62c24e0e781cd565acf83e1
4008930479d911e3235e91f1f9cc4d94c3fad729204698a501b42169a05adf042c
73e09735b98cf8ea48ecf7060c2c95385112ac4f38ae382c4fb6287d2be3c2b07d9
1b0d217cadde6e63a2352a7320b4db2c3c7214e816eb22a24d1c06aede65f38c7
d40357598f7bb74c0

Fig. 12 (**a**) Sobel values of first 60 rows and (**b**) Difference of images due to LSB insertion at the use case

Quantitative and qualitative results

In this section, appraisal metrics regarding the proposed steganography scheme are discussed. More specifically, an assessment of how the algorithm responds based on the evaluation factors regarding steganography presented in the introductory Section, as well as time execution metrics are given.

Capacity

In order to measure the algorithmic capacity, the following conventions are assumed:

- The encrypted form of the '*data*' plus '*authentication data*' is given by the z variable.
- The width of the image by the w variable.
- The height of the image by the h variable.
- The image consists of grayscale pixels.
- '*RONI*' refers to the amount of pixels that consist the RONI
- '*ROI*' refers to the amount of pixels that consist the ROI

Then, as aforementioned in the proposed scheme the RONI consists of $\left\lceil \frac{(z+13)}{\lfloor \frac{w}{2} \rfloor} \right\rceil$ *2 rows or else

$$'RONI' = \left\lceil \frac{(z+13)}{\lfloor \frac{w}{2} \rfloor} \right\rceil *2*w \text{ Pixels.}$$

For reasons of simplicity, the upper and lower bounds are removed from the above equation. Thus,

$$'RONI' = \frac{(z+13)}{\frac{w}{2}} *2*w = 4*z + 52 \text{ Pixels.} \quad (1)$$

Furthermore, if every pixel of the ROI is used during insertion it will consist of 2*z pixels, since the ROI consist of two shares. In addition, since all of the pixels of the image are $w*h$, ROI and RONI must be less or equal to $w*h$.

$$\begin{aligned} &\Rightarrow w*h \geq 'ROI' + 'RONI' \\ &\Rightarrow w*h - 'ROI' - 'RONI' \geq 0 \\ &\Rightarrow w*h - 4*z + 52 - 2*z \geq 0 \\ &\Rightarrow 6*z \leq (-52 + w*h) \end{aligned} \quad (2)$$

In addition if the first and last row of the image is not used during insertion, due to the fact that they have zero gradient magnitude values, Eq. 2 can be rewritten as:

$$\begin{aligned} &\Rightarrow 6*z \leq -52 + w*h - 2*w \\ &\Rightarrow 6*z \leq (h-2)*w - 52 \\ &z \leq \frac{(h-2)*w-52}{6} \end{aligned} \quad (3)$$

If either no encryption or a stream cipher is used, z will be equal (in size) to the '*data*' and '*authentication data*'. Thus, the amount of '*data*' that can be hidden is:

$$'data' \leq \frac{(h-2)*w-52}{6} - 'authentication\ data' \quad (4)$$

For example, based on Eq. 3 for an image of size 512 × 512, the maximum amount of '*encrypted data*' that can be hidden if a stream cipher is used is 43511 bits ≈ 5.3 Kbyte. Furthermore, when a stream cipher is used the total insertion in bits equals to:

$$3*('data' + 'authentication\ data') + 26 \text{ bits} \quad (5)$$

This is because the *'map'* occupies twice the *'encrypted data'* which has a size of $('data' + 'authentication\ data')$ bits. In addition, *'start row'* and '*size of map*' occupy 26 bits. Thus, for an image of size 512 × 512, if SHA-256 is used, then the

'map':
1111110000000000111100001100001111001111110011001111110011111100
1111000000111111000011111100000000110011000000001100110011001100
0011001100001111111001111111100000000000100110000001100111111100
0000110000000011110011111100110100000000000000000000010101001100
1101000100000001000000000001110000110001011111000101010000000...

Fig. 13 Part of Extraction 'map' produced by LSB insertion at the use case

0000001011	0000100110000000	1111110000000000011110…
a	b	c

Fig. 14 'recovery data' inserted using the 2LSB insertion algorithm at the use case: (**a**) 'start row', (**b**) 'size of map', (**c**) 'map'

max amount of *'data'* that can be hidden is equal to (43511 − 256 ∗ 2) = 42999 bits. Furthermore, based on Eq. 5 a total of 3 ∗ (43511) + 26 = 130559 bits can be inserted, which corresponds to $\frac{130559}{512*512} \approx 0.50$ bpp.

Finally, regarding the amount of *'data'* which can be inserted when a block cipher is used, in the worst case an additional block must be deducted. Thus, the following equation occurs

$$'data' \leq \frac{(h-2)*w-52}{6} - 'authentication\ data' - 'block\ size' \quad (6)$$

Robustness

Since the proposed schemes manipulate the LSB or both the LSB and the second LSB of the pixels any modification of the pixels will destroy the hidden data. Nevertheless, since the integrity of the images is of utmost importance even slightly modified images should not be used.

Imperceptibility

In this subsection, the imperceptibility of the stego images is evaluated. More specifically the peak signal-to-noise ratio (PSNR) values are utilized to test visual similarity between the stego and cover images. Furthermore, the stego images are tested against two steganalysis algorithms. In order to have objective results 10 16-bit grayscale medical images (CTs and MRIs) of size 512 × 512 and 2 of size 256 × 256 are used. These images are presented in Table 5. In every image an insertion of 0.25 bits per pixel takes place using randomly generated data through the use of a uniform probability distribution. It must be noted that the calculation of the total amount of insertion (in bits per pixel) which takes place is calculated based on Eq. 7. In Eq. 7 the calculation takes into account all the data inserted except the *'start row'* and *'size of map'*. Furthermore, *'total pixels'* refers to the number of pixels that comprise the image.

$$\frac{3*('data' + 'authentication\ data')}{'total\ pixels'} \quad (7)$$

PSNR calculation

Given a reference (cover) image f and a test (stego) image g, both of size *M*x*N*, PSNR is defined using the mean square error (MSE) as follows:

$$\mathrm{PSNR}(\mathrm{f},\ \mathrm{g}) = 10\log 10\left(\frac{('\max\ value')^2}{\mathrm{MSE}(\mathrm{f},\ \mathrm{g})}\right) \quad (8)$$

Where

$$\mathrm{MSE}(\mathrm{f},\ \mathrm{g}) = \frac{1}{M*N}\sum_{i=1}^{M}\sum_{j=1}^{N}(fij-gij)^2 \quad (9)$$

and '*max value'* the maximum possible pixel value. For example the *'max value'* for 8 bit grayscale images is 2^8-1 = 255.

In Table 6, based on Eqs. 8 and 9 the PSNR and MSE values for every set of images are given. To be more precise every set consists of the images before and after embedding.

Similar measurements in [6] had much lower PSNR values for even smaller insertion of data. In this context, the PSNR values produced are considered more than acceptable.

Resistance versus steganalysis

In order to evaluate the proposed schemes resistance to steganalysis the stego images are tested against two structural LSB detectors. The first is the weighted stego steganalyser described in [28] (referenced in this paper as "WS"), while the second detector is the structural LSB detector described in [29] (referenced in this paper as "TRIPLES"). For both detectors the implementations given by Jessica Fridrich in [30] are used. Both detectors attempt to estimate the length of the hidden message and thus the payload (bits per pixel) that has been inserted into the stego image. The equivalent results for an embedding rate of 0.25 bpp are given in Table 7.

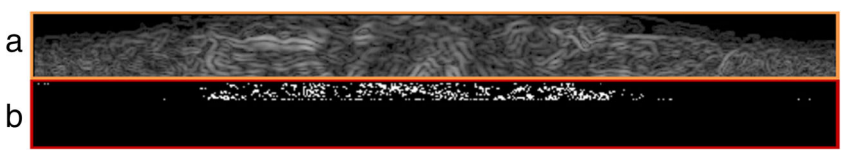


Fig. 15 (**a**) Sobel values of first 60 rows and (**b**) Difference of images due to 2LSB insertion at the use case

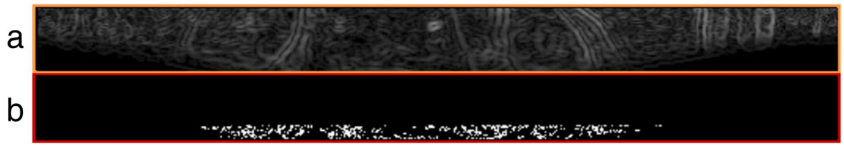


Fig. 16 (**a**) Sobel values of last 60 rows and (**b**) Difference of images due to 2LSB insertion at the use case

In all cases the estimated embedding rate is much smaller than the real one. More specifically in all images except 7, 8, 9, and 11 the estimated embedding rate is below 0.014 while at most images it is below 0.008. The reason why the estimated embedding rate is significantly higher in these images is due to the fact that they contain large black (smooth) areas, which if modified are susceptible to steganalysis. Thus, it can be stated that at least for DICOM images that do not contain a large amount of smooth regions the proposed algorithm has a high resistance against steganalysis.

Privacy/security

The specific evaluation criterion refers to the amount of secret knowledge needed during the algorithm execution. This corresponds to the size of the cryptography key, which depends

Table 5 DICOM images used for testing

Image 01 - MR - 512x512	Image 02 - CT - 512x512	Image 03 - CT - 512x512	Image 04 - MR - 512x512

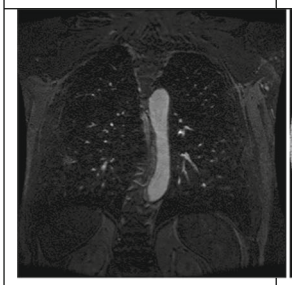
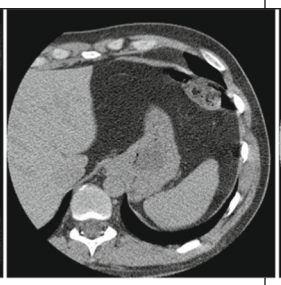
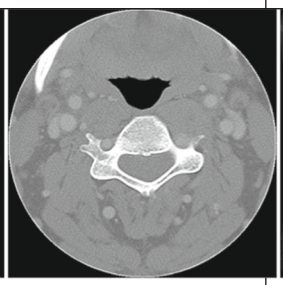

		

Image 05 - MR - 512x512	Image 06 - MR - 512x512	Image 07 - MR - 512x512	Image 08 - CT - 512x512

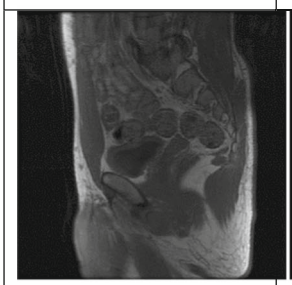
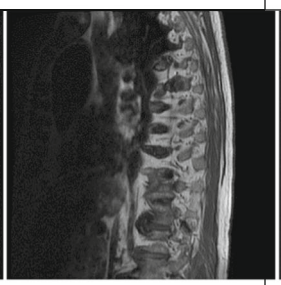
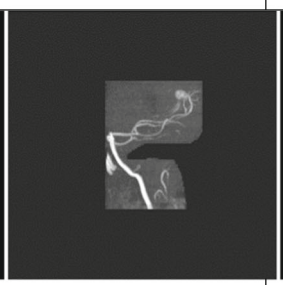
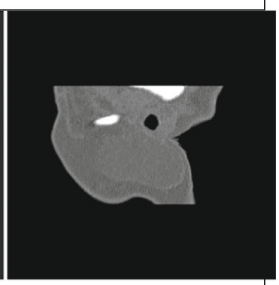

			

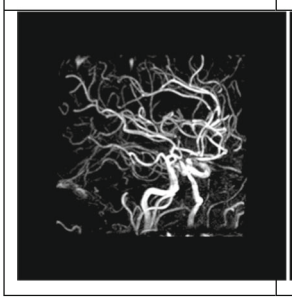
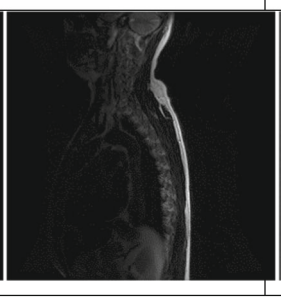
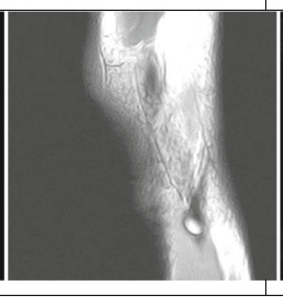
Image 09 - MR - 512x512	Image 10 - MR - 512x512	Image 11 - MR - 256x256	Image 12 - MR - 256x256
			

Table 6 PSNR values of the 12 DICOM images tested for an insertion rate of 0.25 bpp

Image Number	Dimensions of image	MSE	PSNR
01	512 × 512	0.0830	107.1403
02	512 × 512	0.1250	105.3617
03	512 × 512	0.1253	105.3500
04	512 × 512	0.1241	105.3924
05	512 × 512	0.1239	105.3983
06	512 × 512	0.1243	105.3843
07	512 × 512	0.1840	103.6816
08	512 × 512	0.0828	107.1503
09	512 × 512	0.1650	104.1550
10	512 × 512	0.1238	105.4035
11	256 × 256	0.1267	105.3030
12	256 × 256	0.1269	105.2962

on the encryption scheme used. In the experimental results, AES with a key size of 128 bits was selected.

Complexity and execution times

In this subsection, metrics regarding the execution time of the proposed schemes are given. To be more precise, the execution times refer to the insertion and extraction process and they are listed in Table 8 for the evaluated images in an Intel i5-3230 M 2.6GHz CPU. As it is illustrated, for an insertion rate of 0.25 bpp in images of 512 × 512 pixels size, the mean execution time of the insertion scheme is 697.4 milliseconds with a standard deviation of 85.3 milliseconds, while the mean execution time of the extraction scheme is 403.4 milliseconds with a standard deviation of 9.4 milliseconds. Furthermore, for an insertion rate of 0.25 bpp in images of 256 × 256 pixels size the mean execution time of the insertion scheme is 452.5 milliseconds with a standard deviation of 65.8 milliseconds, while the mean execution time of the extraction scheme is 137.5 millisecond with a standard deviation of 4.9 milliseconds.

Evidently, even for an insertion rate of 0.25 bpp, the insertion scheme demands less than 1 s in order to execute while the corresponding time for the extraction scheme is less than 0.5 s. Furthermore, by using more than one physical and logical core the above execution times can be drastically reduced.

The reason the insertion scheme demands more execution time than the extraction scheme relies on the fact that extra calculations, such as the calculation of the image gradient are required during insertion.

Table 7 Estimation of bits per pixel inserted given by the structural detectors in [28] (WS) and [29] (Triples)

Image Number	WS	TRIPLES
01	−0.0021	0.0048
02	0.0038	0.0008
03	0.0035	0.0008
04	0.0054	0.0025
05	0.0068	−0.0127
06	0.0027	−0.0039
07	0.0916	0.0488
08	−0.0030	0.0334
09	0.1085	0.0629
10	0.0076	0.0047
11	0.0594	0.3684
12	0.0095	0.0137

Conclusions and future work

In this paper we proposed and evaluated a steganography algorithm for medical images which provides reversibility of the ROI, as well as data integrity. The presented experimental results proved that the proposed scheme could be used as an efficient steganography scheme, when images with limited smooth areas are used. Furthermore, it was also proven that as far as visual equality (in terms of PSNR) is concerned, excellent results are produced. Finally, since the algorithm is used in spatial domain DICOM images, high capacity insertion is applicable.

The proposed method can be utilized for the exchange of medical images over the Internet and the storage of image data in Cloud infrastructures. Nowadays, cloud computing is recognized as a dominant computing model in IT infrastructures, enabling flexible, ubiquitous, on-demand and cost-effective access to a wide pool of shared resources in all business sectors including healthcare. One of the main concerns in such information systems remains data privacy. Furthermore, sharing health information raises the level of complexity and increases the stakes for issues of data confidentiality and the

Table 8 Execution times of the insertion and extraction scheme during analysis

Image Number	Execution time of insertion scheme (in milliseconds)	Execution time of extraction scheme (in milliseconds)	Total Execution time (in milliseconds)
01	609	390	999
02	733	412	1145
03	749	404	1153
04	671	421	1092
05	764	407	1171
06	858	407	1265
07	577	400	977
08	733	391	1124
09	656	405	1061
10	624	397	1021
11	499	134	633
12	406	141	547

need for robust security measures. The presented methodology exhibited very good results in all aspects of evaluation concerning Capacity, Robustness, Imperceptibility and Privacy. Furthermore execution times and data overheads are low, allowing easy and transparent integration in existing distributed medical information systems.

As described in section 2.1.5, regarding the pixels selected to be modified, there is a likelihood of modification of the pixel with the smallest gradient magnitude. To address this issue, 3 bits per map entry could be used. In this case, the first bit would state if a modification occurred, the second bit in which share the modification transpired and the third bit if a border value was modified. Nevertheless, this was not considered a beneficial choice since it would significantly increase the amount of data inserted, while experimental results proved that this is a rare case. Actually, during our experimentation this case did not appear at all.

When DICOM images with large smooth areas are used, the scheme could be used as a watermarking scheme. More specifically, it can be derived that two watermarks are inserted: Watermark *W1* which is composed of the '*recovery data*' and is inserted in the RONI and *W2* which is composed of both the '*data*' and '*authentication data*' (in an encrypted form) and is inserted in the ROI. Thus, future work involves inserting error control mechanisms in our scheme to make it more robust and subsequently even more suitable for watermarking [6, 8, 31].

References

1. Kagadis, G. C., Kloukinas, C., Moore, K., Philbin, J., Papadimitroulas, P., Alexakos, C., Nagy, P. G., Visvikis, D., and Hendee, W. R., Cloud computing in medical imaging. *Med. Phys.* 40(7):070901, 2013. doi:10.1118/1.4811272.
2. Teng, C., Mitchell, J., Walker, C., Swan, A., Davila, C., Howard, D., and Needham, T, A medical image archive solution in the cloud. In: IEEE international conference on software engineering and service sciences (ICSESS), Beijing, China, pp 431–434, 2010 doi: 10.1109/ICSESS.2010.5552343.
3. Maglogiannis, I., Delakouridis, C., and Kazatzopoulos, L., Enabling collaborative medical diagnosis over the internet via peer-to-peer distribution of electronic health records. *J. Med. Syst.* 30:107–116, 2006. doi:10.1007/s10916-005-7984-1.
4. Nergui, M., Acharya, U. S., Acharya, R., and Yu, W., Reliable and robust transmission and storage techniques for medical images with patient information. *J. Med. Syst.* 34:1129–1139, 2010. doi:10.1007/s10916-009-9332-3.
5. National Electrical Manufacturers Association. Digital imaging and communications in medicine (DICOM). http://dicom.nema.org/standard.html. Accessed 16 December 2015, 2015.
6. Al-Qershi, O. M., and Khoo, B. E., Authentication and data hiding using a hybrid ROI-based watermarking scheme for DICOM images. *J. Digit. Imaging* 24:114–125, 2011. doi:10.1007/s10278-009-9253-1.
7. Guo, X., and Zhuang, T. G., A region-based lossless watermarking scheme for enhancing security of medical data. *J. Digit. Imaging* 22:53–64, 2007. doi:10.1007/s10278-007-9043-6.
8. Giakoumaki, A., Pavlopoulos, S., and Koutsouris, D., Multiple image watermarking applied to health information management. *IEEE Trans. Inf. Technol. Biomed.* 10(4):722–732, 2006. doi:10.1109/TITB.2006.875655.
9. Tian, J., Reversible data embedding using a difference expansion. *IEEE Trans. Circ. Syst. Video* 13(8):890–896, 2003. doi:10.1109/TCSVT.2003.815962.
10. Coatrieux, G., Lecornu, L., Roux. Ch., and Sankur, B, *A review of image watermarking applications in healthcare*. In: Proc. of 28thAnnual International Conference Engineering in Medicine and Biology Society, EMBS '06. New York: IEEE, pp. 4691–4694, 2006. doi: 10.1109/IEMBS.2006.259305.
11. Wang, H., and Wang, S., Cyber warfare: Steganography vs. steganalysis. *Commun. ACM* 47(10):76–82, 2004. doi:10.1145/1022594.1022597.
12. Mielikainen, J., LSB matching revisited. *IEEE Signal Proc. Lett.* 13(5):285–287, 2006. doi:10.1109/LSP.2006.870357.
13. Chan, C. K., and Cheng, L. M., Hiding data in images by simple LSB substitution. *Pattern Recogn.* 37(3):469–474, 2004. doi:10.1016/j.patcog.2003.08.007.
14. Zhang, X., and Wang, S., Efficient steganographic embedding by exploiting modification direction. *IEEE Commun. Lett.* 10(11):781–783, 2006. doi:10.1109/LCOMM.2006.060863.
15. Chao, R. M., Wu, H. C., Lee, C. C., and Chu, Y. P., A novel image data hiding scheme with diamond encoding. *EURASIP J. Inf. Secur.* 2009:1–9, 2009. doi:10.1155/2009/658047.
16. Hong, W., and Chen, T., A novel data embedding method using adaptive pixel pair matching. *IEEE Trans. Inf. Forensic Secur.* 7:176–184, 2012. doi:10.1109/TIFS.2011.2155062.
17. Wu, D. C., and Tsai, W. H., A steganographic method for images by pixel-value differencing. *Pattern Recogn. Lett.* 24(9–10):1613–1626, 2003. doi:10.1016/S0167-8655(02)00402-6.
18. Yang, C. H., Weng, C. Y., Wang, S. J., and Sun, H. M., Adaptive data hiding in edge areas of images with spatial LSB domain systems. *IEEE Trans. Inf. Forensic Secur.* 3(3):488–497, 2008. doi:10.1109/TIFS.2008.926097.
19. Luo, W., Huang, F., and Huang, J., Edge adaptive image steganography based on LSB matching revisited. *IEEE Trans Inf Forensics* 5(2):201–214, 2010. doi:10.1109/TIFS.2010.2041812.
20. Hong, W., Chen, T. S., and Luo, C. W., Data embedding using pixel value differencing and diamond encoding with multiple-base notational system. *J Syst. Softw.* 85(5):1166–1175, 2012. doi:10.1016/j.jss.2011.12.045.
21. Liu, J., Tang, G., and Sun, Y., A secure steganography for privacy protection in healthcare system. *J. Med. Syst.* 37:9918, 2013. doi:10.1007/s10916-012-9918-z.
22. Muhammad, K., Sajjad, M., and Baik, S. W., Dual-level security based cyclic18 steganographic method and its application for secure transmission of keyframes during wireless capsule endoscopy. *J. Med. Syst.* 40:114, 2016. doi:10.1007/s10916-016-0473-x.
23. Yuan, H. D., Secret sharing with multi-cover adaptive steganography. *Inf. Sci.* 254:197–212, 2014. doi:10.1016/j.ins.2013.08.012.
24. Aryanto, K. Y., Oudkerk, M., and van Ooijen, P. M., Free DICOM de-identification tools in clinical research: Functioning and safety of patient privacy. *Eur. Radiol.* 25(12):3685–3695, 2015. doi:10.1007/s00330-015-3794-0.
25. Vairalkar, M. K., and Nimbhorkar, S. U., Edge detection of images using Sobel operator. *Int. J. Emerg. Technol. Adv. Eng.* 2:291–293, 2012. ISSN 2250–2459.
26. Daemen, J., and Rijmen, V., *The design of rijndael: AES - the advanced encryption standard*. Springer, Berlin Germany, 2002.
27. Thakur, J., and Kumar, N., DES, AES and blowfish: Symmetric key cryptography algorithms simulation based performance analysis.

Int. J. Emerg. Technol. Adv. Eng. 1(2):6–12, 2011. ISSN 2250–2459.

28. Ker, A. D., and Böhme, R., Revisiting weighted stego-image steganalysis. In: Delp, E. J., Wong, P. W., Dittmann, J., Memon, N. D. (Ed.) *Security, Forensics, Steganography, and Watermarking of Multimedia Contents X (Proc of SPIE)*, Volume 6819, San Jose, CA, 2008. doi: 10.1117/12.766820.
29. Ker, A. D., A general framework for the structural steganalysis of LSB replacement. In: Barni, M., Herrera, J., Katzenbeisser, S., and Pérez-González, F. (Eds.), *Lect notes comput Sc, 7th international workshop*, vol. 3727. Springer, Berlin, pp. 296–311, 2005. doi:10.1007/11558859_22.
30. Jessica Fridrich. Structural LSB detectors. http://dde.binghamton.edu/download/structural_lsb_detectors/. Accessed 16 December 2015.
31. Nayak, J., Bhat, P. S., Rajendra Acharya, U., and Sathish Kumar, M., Efficient storage and transmission of digital fundus images with patient information using reversible watermarking technique and error control codes. *J. Med. Syst.* 33(3):163–171, 2009. doi:10.1007/s10916-008-9176-2.