Digital Steganography for Images, assisted by Cryptography

Mauricio Castaño Uribe
Universidad EAFIT
Medellín, Colombia

José Miguel Gil Valencia
Universidad EAFIT
Medellín, Colombia

Juliana Henao Arroyave
Universidad EAFIT
Medellín, Colombia

Isabel Urrego Gómez
Universidad EAFIT
Medellín, Colombia

Gerónimo Zuluaga Londoño
Universidad EAFIT
Medellín, Colombia

November 2019

Abstract

The current research was carried out to present the fundamental characteristics of cryptography techniques and steganography techniques in order to apply them together for a better accomplishment of the aim of keeping a conversation secret. With that in mind, first it was thoroughly investigated about the different types of cryptography (public key and private key) and the methods to apply this techniques, next it was implemented the private key methods in MATLAB. Then, it was researched about the fundamentals of steganography and focused in three techniques for hiding messages in digital images, which are: Less significant bit for RGB images, another implementation to RGB pictures and less significant bit in Grey-scale images; this techniques were implemented in MATLAB as well. After a fully comparison of the three previous methods with each other, finally the cryptography algorithms and the steganography algorithms were implemented together seeking to meet the proposed objective.

Keywords— Steganography, cryptography, secret message, security, LSB, RGB, Grey-Scale, images

Contents

1 Introduction

In the data and globalization age, the information interchange is a high significance topic, which great researches and specializations have arisen from, since is not always easy to make these interchanges in a secure and efficient way. Cryptography may be one of the most used methods for protecting information, however, it is not the only present method with this objective. Steganography is one of those not frequently mentioned methods, and is the one which this investigation will focus on.
On first place it has to be highlighted the principal difference between both techniques, which is that in steganography, a person or group of people, unrelated to the emitter and the receptor of the message, are not aware of the secret hidden in it; as opposed to cryptography, in which it is not always that way, because it is not important that the message after being encrypted has any sense. The principal objective of steganography is that in spite of a third person having the channel in which the secret message will be send, this person can not know that there has been a manipulation of it.
Even though being in essence different methods, the objective may be the same: hiding and protecting information; as far as it will be demonstrated bellow, it is possible to increase security using both techniques together.

2 Cryptography

There are two types of encryption, public key and private key, different methods of each class have been developed, but in this case study we will only consider some such as ECB, CBC, CFB and OFB that are ways of encrypting working with key private, and by public key RSA and Difie-Hellman
It begins explaining from the simplest to the most complex method, to emphasize the differences each time it is staggered and becomes safer, and which method best fits the steganography.

Then cryptography will be defined in a lax way, this has been worked from the beginning as the way to hide messages through encryption and coding that are responsible for modifying the linguistic representations of the message that you want to hide, making this message in a way or another difficult to understand, this tool allows secure and confidential communication, which makes this tool so popular because at this time of the digital era, secure channels or secure forms of communication are needed. Having this definition of cryptography you can pass to the different encryption methods.

We have the ECB (Electronic Code Book Mode) this method is the simplest of all it is to encrypt the message by separating it by blocks and encrypting each block separately, it is a quick method by having the facility of dividing the main message into blocks and encrypt independently but has many disadvantages such as that the encrypting the same message several times, always will have the same result, you can perform dictionary attacks, in an attack can be eliminated blocks and can not account for, unless you have the original message and block capture. The first method is one of the most simple to implement and gives a view of what is cryptography, but has too many weak points.

In the second we have the CBC (Cipher Block Chaining Mode) is an extension of the ECB method with some added security when encrypting by blocks as in the previous method, adding that when passing from block to encrypt the previous encryption of the block is used to encrypt the block current. An initiation vector is also used to encrypt the first block and after using the encryption that exits the block for the next one, the disadvantages of this method is that it is encrypted sequentially and parts of the message can be captured.

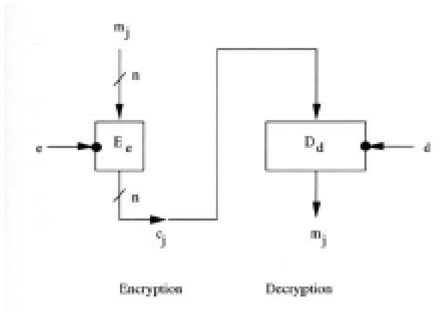


Figure 1: Electronic Code Book Mode [2]

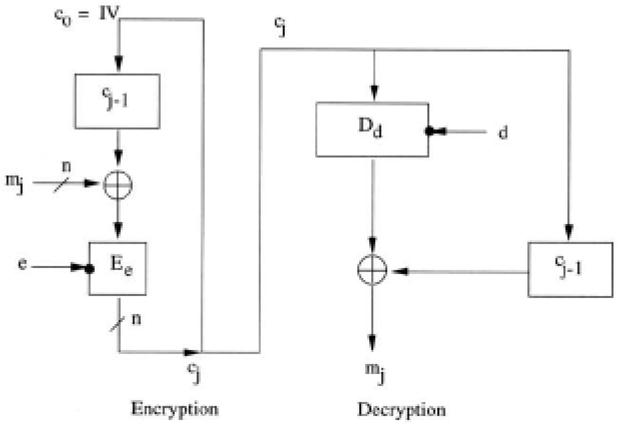


Figure 2: Cipher Block Chaining Mode [2]

Thirdly we have CFB (Cipher Feedback Mode) is similar to the OFB method - explained later - in terms of producing a Keystream keys, when encrypting the last block of encryption, it also uses an initiation vector like the CBC method, this method allows make a counter at the time of encryption and facilitate decryption in parallel, this method by having the counter makes it easy to know if the message has been altered in terms of block theft, and having a Keystream makes it more complex to decrypt the stolen block.

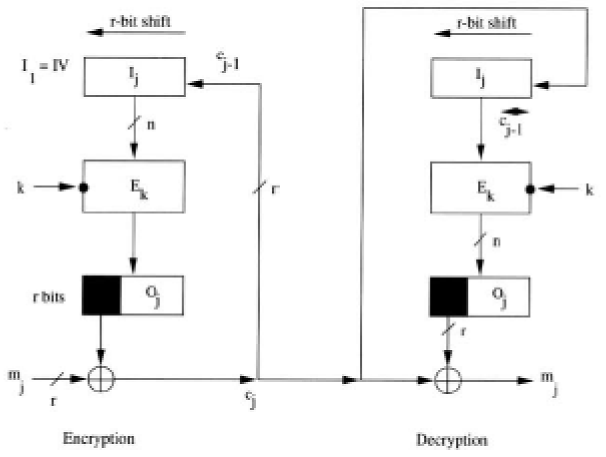


Figure 3: Cipher Feedback Mode [2]

Finally , of the private key methods to work we have the OFB (Output Feedback Mode) is a flow encryption that uses Keystreams to encrypt each block and these Keystrims are generated with each encryption of the previous block, advantage of starting the encryption with a random vector, despite being one of the safest in its class has certain disadvantages such as recalculating the Keystream, not decrypted in parallel and performs a counter.

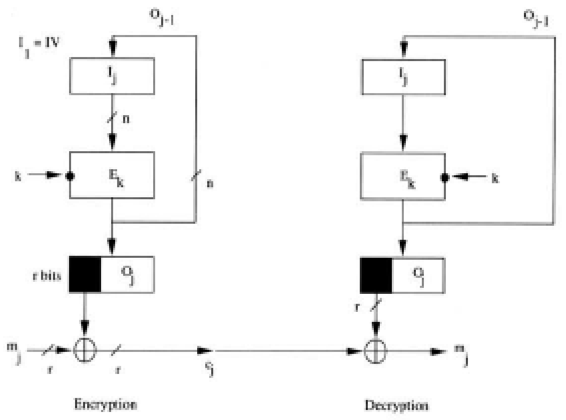


Figure 4: Output Feedback Mode [2]

Turning to the public key methods, it can be said that they are more secure than those of the private key since the information thief will never have the private keys of the sender or receiver, and with these keys it is the only way to know the true message in a considerably short time while the message is being sent.

On the part of RSA , the message is encrypted and signed, a problem of factorization of whole numbers is generated at the time of performing operations to decrypt, the messages are represented by numbers, and the operations to encrypt are based on two large random prime numbers of order 10^{200} and the personal keys are made with prime numbers, the disadvantage of this model is the quantum computation of solving operations.

The Difie-Hellman method is a method to share private keys to perform encryption is based on the fact that each one chooses a private number and in common two numbers are chosen, one that is any cousin P and any generator G, through private numbers and the public are carried out equal operations on both sides to calculate the public keys and the result of these operations is a number that is sent to the other , and through the private keys and the inverse functions an identical secret key is obtained. [2]

3 Steganography

Steganography is usually defined as the art of hiding the presence of a secret message at all. This is due to the fact that since ancient Greek people have been using many creative techniques to send secret messages without being discovered. A remarkable example of this is the Herodotus story about a slave sent by master to the Ionian city of Miletus with a secret message tattooed on his scalp. After tattooing, the slave grew his hair back in order to conceal the message. He then journeyed to Miletus and, upon arriving, shaved his head to reveal the message to Aristagoras. The message encouraged Aristagoras to start a revolt against the Persian king [3].

With the computer revolution and creation of internet, steganography has expanded to digital steganography, and now there are many ways to produce a steganographic product. Today, cryptography is a very common way to hide messages and is a very advanced and refined field, but an encrypted message is very obvious and anyone can clearly see there is a secret communication. On the other hand, the purpose of steganography is to leave no trace of any message. However, the steganography techniques are not as developed as cryptography techniques, but it is still an alternative and complementary tool for keeping a secret message.[4] Besides, the study of steganography is a relevant topic, not only for the embedding of a message but also for the detection and extraction of one, since these techniques have been suspected as a possible way to exchange information about terrorist attacks, this process is called steganalysis.

Therefore, to fulfill the purpose of covering secret communication, the main factor is the undetectability, so no algorithm can say if a picture contains a hidden message.

In the steganography process there are three factors which take action in, the emitter, the receptor and the channel (the channel could be a person). The following figure (figure 1) is the model for the scheme of a secret communication using steganography. The embedding function changes depending of the technique used.

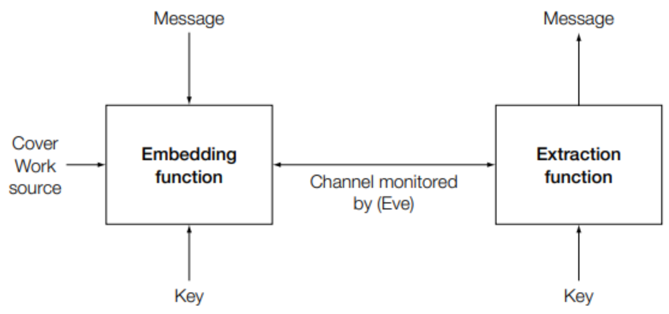


Figure 5: Steganographic embedding scheme.

In order to control the incorporation of the message in a image there is a secret key which is known only by the emitter and the receptor, this key may be just knowing which method was used, or it can also be a seed for a pseudo-random number to know where is the message. In the case of digital steganography the channel is usually via internet and maybe some people (a warden) can see the picture but they can not know there is a communication hidden, the channel properties are defined by the warden. The warden can be a passive one, an active one and a malicious one; this depends on how much he can modify the message. However, for more security the selection of the channel is a topic to take care of, there are techniques for this selection in proposed by Ross Anderson, Fabien A. Petitcolas (1998) [1]

Below are some techniques for hiding messages in digital images, there are many techniques developed, in this document we will deal with three of them.

4 LSB implementation in RGB images

An image in a computer is saved as a Matrix in which every position saves the information of a single Pixel. The matrix is usually three-dimensional (I x j x k) and its respective sizes depend on the height, length and color model of the image.

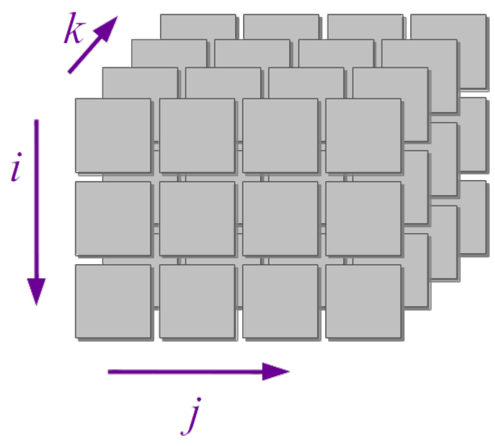


Figure 6: Matrix of an Image

In this case we will talk about an LSB implementation for images with the color model RGB. Therefore, k will be equal 3. This means, that the image can be seen as three layers of I x j matrices and each layer will save the information of each pixel for the weight of the colors: red, green and blue. Here is an illustrative example of a matrix that represents an RGB image: [5]

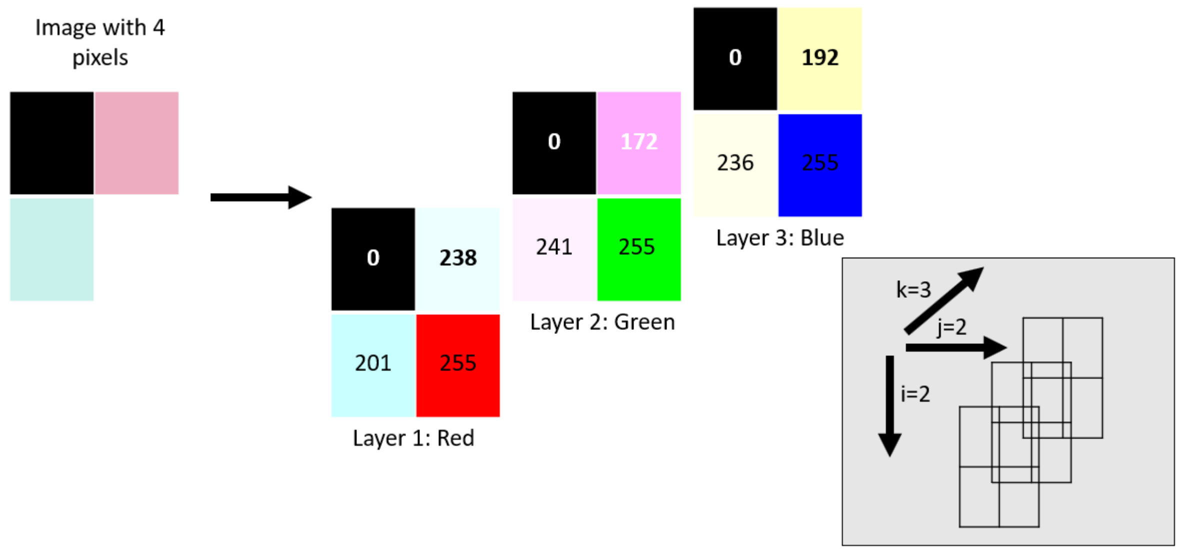


Figure 7: Matricial Representation of an Image

4.1 The embedding algorithm (Intuition)

Now that we understand the computational representation of an image it is time to introduce the concept of LSB. The idea of hiding the information in the least significant bit is to hide each character of the message in the last bits of an A_{ijk} of the matrix. In order to do that is important to mark that the numbers saved in the matrix are written in binary code and since they can go from 0 to 255 the binary representation has a range 00000000-11111111. Additionally, the ASCII code also assigns values that have no longer than 8 digits and this is allows the hidden message to fit perfectly in the least significant bits.

Lets explain the LSB algorithm intuitively with an example based on the image 8. The algorithm is based on the ideas proposed in two different papers: *Enhanced Least Significant Bit algorithm For Image Steganography* [6] and *An Improved LSB Based Steganography Technique for RGB Color Images* [7] . However, there were problems detected in those methods because in them, they used the fact that characters in its binary representation have a maximum length of 7 and that does not count the extension to special characters. Therefore, we extended the algorithm for those cases, because the special characters will be used while combining steganography and cryptography.

For this algorithm, the numbers saved on each pixel will be shown on their binary representation as follows:

Notice that the numbers were separated in blocks of three and we added a 0 before each number. The last block for each number are the least significant bits, because those digits are the ones that allow us to change its values without changing the number itself the least as possible. For example, in the position A_{121} the number is 11101110 and with the additional 0 and the block separation we have 011 101 110 meaning that 110 are the least significant bits. Now we are going to explain the process of message embedding. The number of characters we can hide in this image is 4 because we have 4 pixels. In this case we want to hide the word "love".

Character	ASCII	Binary Representation
l	108	001 101 100
o	111	001 101 111
v	118	001 110 110
e	101	001 100 101

Using the binary representation of each character we only have to save each block of a character in

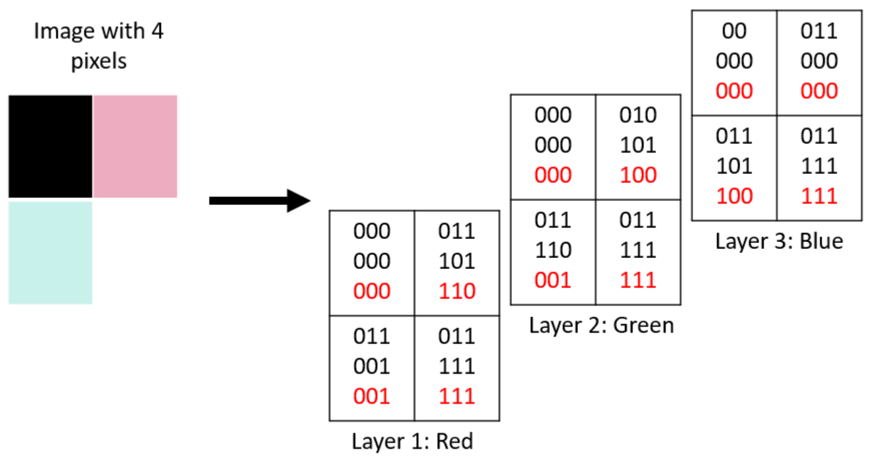


Figure 8: Binary Values of the Image

the same position but in the 3 different layers. For example "l" will be stored in A_{111}, A_{112} and A_{113}. Meaning that those positions will be changed as follows:

Position	Before	After
A_{111}	000 000 000	000 000 001
A_{112}	000 000 000	000 000 101
A_{113}	000 000 000	000 000 100

Applying those changes to every position of the matrix we obtain the following:

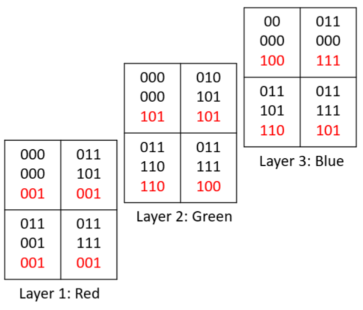


Figure 9: Binary Representation with Embedded Message

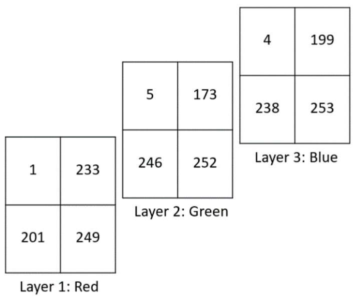


Figure 10: Decimal Representation with Embedded Message

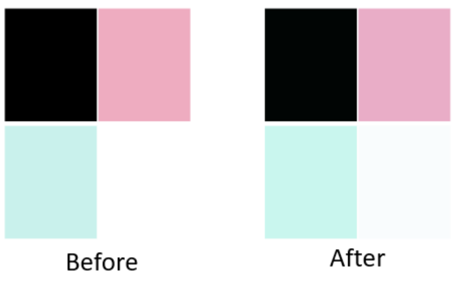


Figure 11: Result

The success of steganography is being able to send the message so that another person does not know that something is hidden. Therefore, is important that the only thing sent by the emitter is the image.

Additionally, we added another safety feature to the embedding and also extraction process: hiding the message in pseudo-random pixels of the image. We say pseudo-random because those values are known by the emitter and receptor but would be difficult for a foreigner to find out the order and the pixels, because there is no pattern on them. In order to do that, we added two vectors to the input of the codes, one for the x values and another one for y values and this is the embedding algorithm. [5]

```
LSB in RGB: Embedding Algorithm
Input image,message,X,Y
Initialize: img ← $image, m \leftarrow message(ASCII)inbinary$
for each bit of the message B(i) do
    for each layer of img I(k) do
        1. Add 0s to the binary representation of the $i^{\text{th}}$ character of the message
        until it has 9 digits
        2. Split each letter of the message into three blocks of three digits
        3. Save the $k^{\text{th}}$ block in the $k^{\text{th}}$ layer of the pixel in position X(i) and Y(i)
        and in the last three bits (the least significant ones)
    end for
end for
stego-image ← $image$
return stego-image
```

The extraction process follows the same idea in the reverse way, the receptor receives the image and goes through the pixels that contain the information and for each pixel builds the hidden character by extracting and joining the information stored in the least significant bits. This is the algorithm:

```
LSB in RGB: Extraction Algorithm
Input stego-image,X,Y
Initialize: img ← $image, m \leftarrow message\quad (ASCII)\quad in\quad binary$
for each position of the vectors X(i),Y(i) do
    str ← $(empty\quad string)$
    message ← $(empty\quad string)$
    for each layer of img I(k) do
        1. Take the number in the position X(i), Y(i),k of the image and
        take its binary representation
        2. Add 0's to the binary representation until it has 9 digits
        3. Extract the digits in positions 7-9 and concatenate them on the right with str
    end for
    auxStr ← $character\quad with\quad binary\quad representation\quad str$
    message ← $message + str$
end for
return message
```

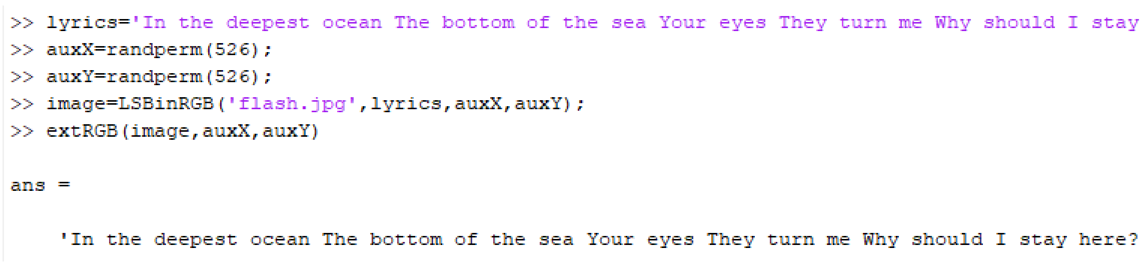
4.2 Example

To ilustrate the embeddig and extraction process with a real implementation, the following text was used:

In the deepest ocean
The bottom of the sea
Your eyes
They turn me
Why should I stay here?
Why should I stay?
I would be crazy not to follow
Follow where you lead
Your eyes
They turn me
Turn me on to phantoms
I follow to the edge of the earth
And fall off

Everybody leaves
If they get the chance
And this is my chance
I get eaten by the worms
Weird fishes
Get towed by the worms
Weird fishes
Weird fishes
Weird fishes
I will hit the bottom
Hit the bottom and escape
Escape
I will hit the bottom
Hit the bottom and escape
Escape
(Radiohead: "Weird Fishes/Arpeggi")

Both algorithms were implemented in Matlab and named as two different functions: LSBinRGB and extRGB. The first one is the one that hides the message in the picture with the given key (the positions represented by vectors X and Y) while the second one extracts the message. The idea is that the returning image of the first function is the only thing that the emitter sends to the receptor and the key is something they previoulsy agreed and had. This way, it will be more difficult for a foreign person to suspect about a hidden message inside the picture.



```
>> lyrics='In the deepest ocean The bottom of the sea Your eyes They turn me Why should I stay
>> auxX=randperm(526);
>> auxY=randperm(526);
>> image=LSBinRGB('flash.jpg',lyrics,auxX,auxY);
>> extRGB(image,auxX,auxY)

ans =

    'In the deepest ocean The bottom of the sea Your eyes They turn me Why should I stay here?
```

Figure 12: LSB in RGB

After running the commands shown in Figure 12, is important to compare the original image with the stego-image to finally demonstrate that the changes are not visible to the human eye. Moreover, the fact that the changes are subtle adds value to the effectiveness of steganography since there are no visual clues that might expose the use of steganography inside the picture. (See figure 13)

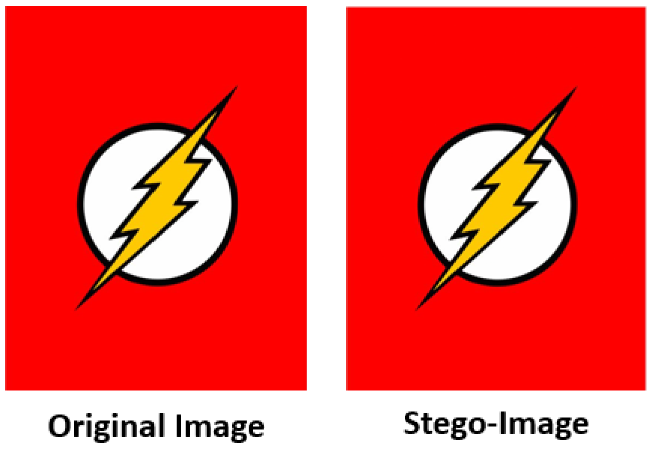


Figure 13: Result of LSB in RGB

5 Another Implementation of LSB in RGB

An alternative way of steganography in RGB images is hiding the whole information in the last layer, because it can be seen as the least significant one.

In spite of one pixel containing the whole information about a single character, the elements stored in the last layer (blue) for each position will be just single bits of the message on its binary ASCII representation. Therefore, the amount of blue will not have significant changes, maximum one unit. Besides, since the amount of blue is only a percentage of the whole pixel, the stego-image will look exactly like the original one.

To explain the situation there is an example: (Figure 14)

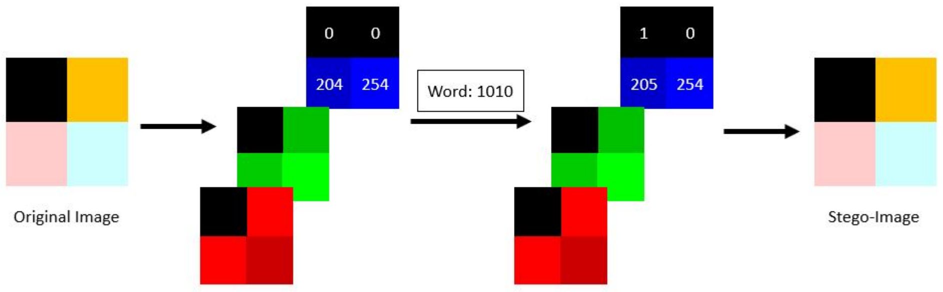


Figure 14: Intuition of LSB in Blue Layer

Here, the image has only four pixels and only the weights stored in the blue layer are shown. The reason for that is because those are the ones that will suffer a change. The embedding process separates the word in its four bits and saves one bit for each pixel: Black is the first bit, the orange pixel the second, the rose one is the third and at last the blue pixel will save the fourth bit. In particular, the orange and blue pixels will not change at all since their bit values were zeros. At last, the stego-image looks exactly like the orignial one because the changes were not significant.

This implementation of LSB is very succesfull because the image can for sure go unnoticed through a channel. For this instance, the layer used was the blue one but the same process will be also very effective using one of the other two. Furthermore, in a future work there is another possibility that can be explored and is saving the information in one bit but in the layer that has the minor value.

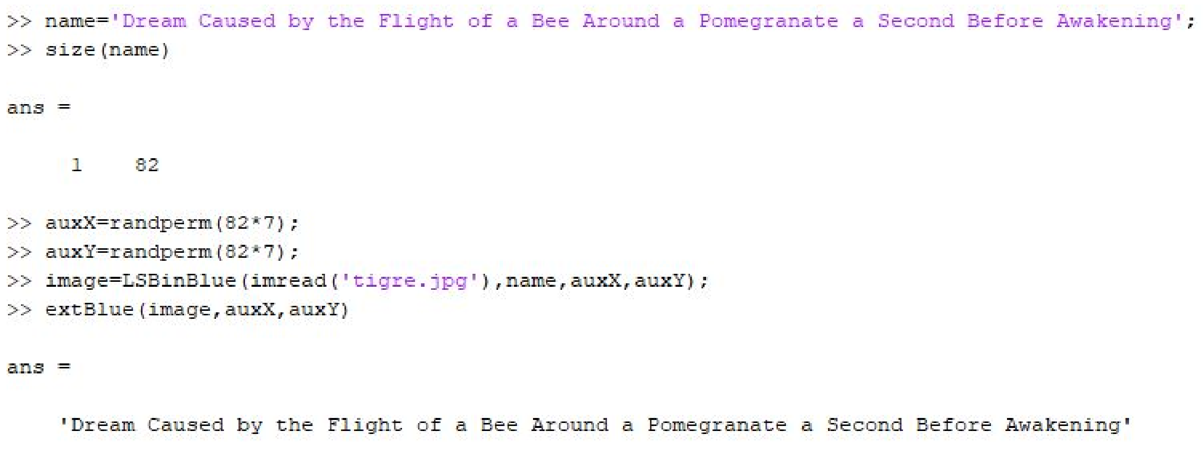
```
**LSB in Blue Layer: Embedding Algorithm**
Input image,message,X,Y
Initialize: img ← $blue\ layer\ of\ image, m \leftarrow message\ (ASCII)\ in\ binary$
for each row of img i
for each column of img j
if $img_{ij}\%2 == 1$ then $img_{ij} \leftarrow img_{ij} - 1$
end for
for each position of the vectors X(i),Y(i) do
if $m_i == 1$ then $img_{X(i),Y(i)} \leftarrow img_{X(i),Y(i)} + 1$
end for
stego-image ← $layers\ 1\ and\ 2\ from\ image\ and\ 3\ from\ img$
return stego-image
```

In the extraction process there is something that must be taken into account: a single character is a string of 7 bits but those bits are not hidden together. Therefore, first the algorithm collects all the bits and after that it separates the whole string into groups of seven and then it converts each group into its respective character.

```
LSB in Blue Layer: Extracting Algorithm
Input image,X,Y
Initialize: img $\leftarrow$ $blue\ \ layer\ \ of\ \ image, s \leftarrow empty\ \ string$
message $\leftarrow$ $empty\ \ string$
for each element of X i
    num $\leftarrow$ $img_{X(i)Y(i)}$
    num2 $\leftarrow$ $num\%2$
    nstr $\leftarrow$ $string\ \ version\ \ of\ \ num2$
    s $\leftarrow$ $s + nstr$
end for
i $\leftarrow 1$
while size of s $> 0$ do
    str2 $\leftarrow$ $substring\ \ from\ \ 1-7\ \ of\ \ s$
    str $\leftarrow$ $substring\ \ after\ \ 7^{\text{th}}$ position of s
    aux $\leftarrow$ $character\ \ with\ \ binary\ \ representation\ \ str2$
    message $\leftarrow$ $message + aux$
    i $\leftarrow$ $i + 1$
end while
return message
```

5.1 Example

This version of LSB was again implemented in Matlab as two separated functions called LSBinBlue as the embedding process and extBlue for the extraction. To try both functions the message used is the name of one of the most famous paintings of the spanish artist Salvador Dali: Dream Caused by the Flight of a Bee Around a Pomegranate a Second Before Awakening. Addiotionally, the name was hidden inside the painting itself by writing the respective commands. (Figures 15 and 16)



```
>> name='Dream Caused by the Flight of a Bee Around a Pomegranate a Second Before Awakening';
>> size(name)

ans =

     1    82

>> auxX=randperm(82*7);
>> auxY=randperm(82*7);
>> image=LSBinBlue(imread('tigre.jpg'),name,auxX,auxY);
>> extBlue(image,auxX,auxY)

ans =

    'Dream Caused by the Flight of a Bee Around a Pomegranate a Second Before Awakening'
```

Figure 15: LSB in RGB (Blue layer)

6 LSB in Grey-scale images

Understanding the matrix representation of images, the concept of grey-scale (GS) images follows the same idea as RGB images. However, in this case there is only one layer and the number saved in each position represents the amount of light of that single pixel. For example, 0 is black because it has no light whereas 255 is the numerical representation of white because is 100% light.

In particular, it is noticed that the matrix of a Grey-scale image is much simpler than RGB images but it hinders the hiding of information and that is why it requires a different implementation of LSB.

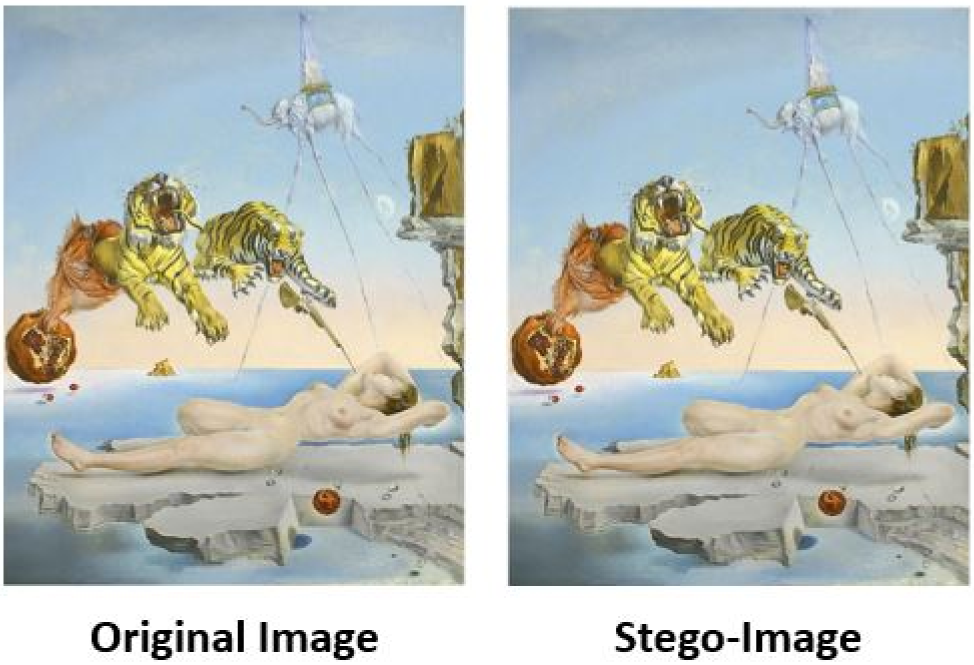


Figure 16: Result LSB in RGB (Blue layer)

To illustrate how this algorithm works in this type of images, suppose we have an image I = $\begin{vmatrix} 203 & 0 \\ 61 & 124 \end{vmatrix}$ and a word W=0011 that we want to hide successfully in the image. Additionally, an auxiliar matrix is needed to determinate the order in which the bits of W are going to be saved, in this particular case we will use A = $\begin{vmatrix} 3 & 2 \\ 4 & 1 \end{vmatrix}$ as the auxiliar matrix. (Notice that A and I must have the same size)

The hiding process takes for example the first element of I (I_{11}) and looks for the value in correspondent position of aux: $A_{11} = 3$. By doing that, we notice that the third bit of W is going to be stored on the position (I_{11}) and we just need to add that bit to the number. In that order, (I_{11}) goes from 203 to 204. By following the same steps with the other (I_{ij}), the first bit of W will be on (I_{22}), the second on (I_{12}) and the fourth on (I_{21}). Finally, the remaining stego-image will be $I_s = \begin{vmatrix} 204 & 0 \\ 62 & 124 \end{vmatrix}$ [8]

A visual representation of those matrices is shown in the following image:

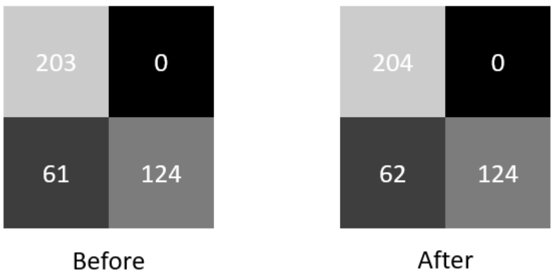


Figure 17: Result

Altough the some of the colors were changed, that changes the number maximum in one unity and that is why those changes are not visible.

This method seems to be very effective, since it just needs the binary representation of the message, what in comparison to the previous one represents a less number of computational operations because it needed the whole matrix in base two.

However, while building the algorithm three main problems were identified:
1. The person who receives I_s can not know if either the pixel was changed or not without having the original image
2. This method adds one to the value in the matrix if the bit that the pixel must save is a 1. Unfortunately, it fails in the case of having a white pixel that should save a 1, since 255 is the maximum value for light intensity.
3. The idea of an auxiliar matrix limitates the possible messages to only the ones that occupy the whole matrix

To deal with these situations the following solution was used while hiding the message:
1. Initialize the image by turning all the odd pixel values into their previous even one. For example $I = 5$ changes to $I = 4$. By doing that all the numbers stored in the matrix will end in 0 on its binary representation.
2. Add 1 to the pixel's value if the bit you want to hide there is a 1. Otherwise keep it the same. That way you fix the problems previously mentioned. The person that receives I_s identifies easily changes in the pixels because even numbers will represent 0's and odd ones will be 1's. Besides, the problem with the white pixel will also be solved.
Additionally, instead of using an auxiliar matrix, the key will be the same as in the previous implementations of LSB, two vectors of xy coordinates.

And that is how the following algorithms were built.

```
LSB in GS: Embedding Algorithm
Input image,message,X,Y
Initialize: img ← $image, m \leftarrow message \quad (ASCII) \quad in \quad binary$
for each row of the img i
for each column of the img j
if $img_{ij}\%2 == 1$ then $img_{ij} \leftarrow img_{ij} - 1$
end for
for each position of the vectors X(i),Y(i) do
if $m_i == 1$ then $img_{X(i),Y(i)} \leftarrow img_{X(i),Y(i)} + 1$
end for
stego-image ← $image$
return stego-image
```

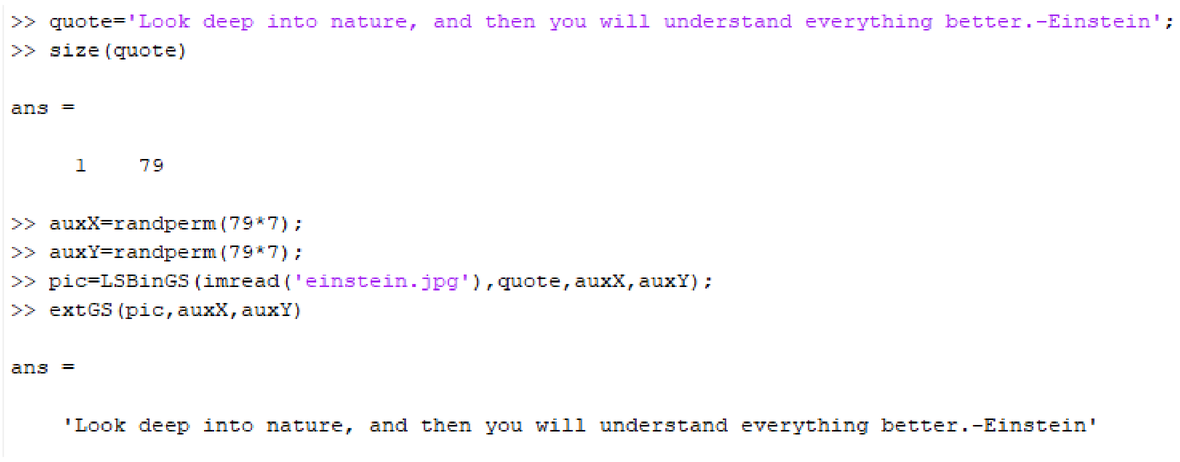
```
LSB in GS: Extracting Algorithm
Input image,X,Y
Initialize: img ← $image, s \leftarrow empty \quad string, message \leftarrow empty \quad string$
for each element of X i
num ← $img_{X(i)Y(i)}$
num2 ← $num\%2$
nstr ← $string \quad version \quad of \quad num2$
s ← $s + nstr$
end for
i ← 1
while size of s > 0 do
str2 ← $substring \quad from \quad 1 - 7 \quad of \quad s$
str ← $substring \quad after \quad 7^{\text{th}}$ position of s
aux ← $character \quad with \quad binary \quad representation \quad str2$
message ← $message + aux$
i ← $i + 1$
end while
return message
```

6.1 Example

In this case the text used is one of Einstein's most famous quotes:

"Look deep into nature, and then you will understand everything better."
-Einstein

The two functions implemented in Matlab were named LSBinGS for the embedding function and extGS for the extraction one. The requirment for the picture is using a greyscale image that has only two dimensions and for the ilustrating example we used a picture of Albert Einstein himself, probably the most famous one. The running test can be seen in figure 18 and the result in figure 19.



```
>> quote='Look deep into nature, and then you will understand everything better.-Einstein';
>> size(quote)

ans =

     1    79

>> auxX=randperm(79*7);
>> auxY=randperm(79*7);
>> pic=LSBinGS(imread('einstein.jpg'),quote,auxX,auxY);
>> extGS(pic,auxX,auxY)

ans =

    'Look deep into nature, and then you will understand everything better.-Einstein'
```

Figure 18: LSB in Greyscale

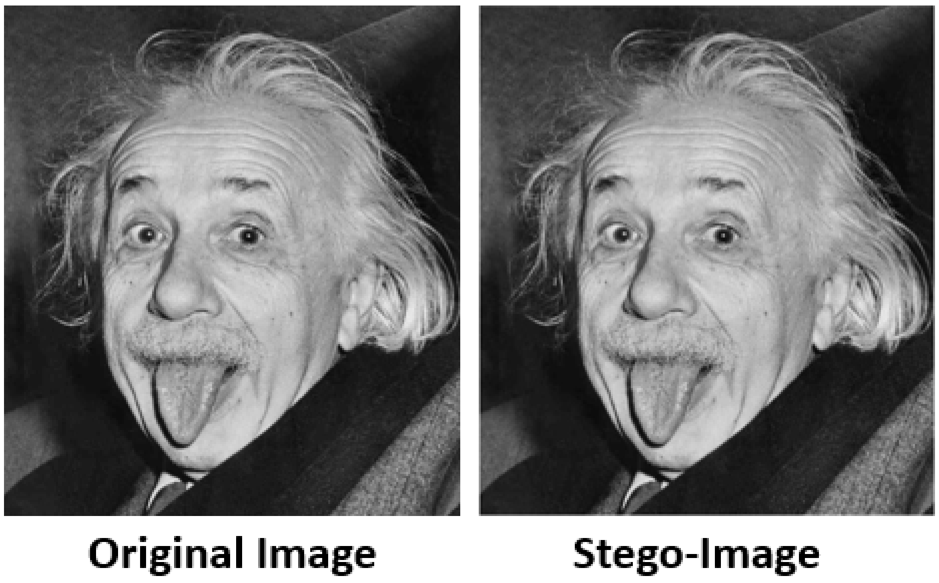


Figure 19: Result of LSB in Greyscale

7 Comparison of the LSB implementations

To compare the three implementations previously mentioned, different aspects were taken into account such as the image used, the message, the stego-images and the algorithmic complexity. The following tables show the conclusions made.

Aspect	LSB in RGB	LSB in Blue Layer	LSB in GS
Image used for embedding process	This algorithm had the only restriction of requiring an RGB image. In the proofs made there were no needs to look for another picture, but it might be some cases were the number of characters of the message exceeds the number of pixels.	This method requires an RGB image. Additionally, while doing some running tests, there were some images where the hiding process could not be made because the length of the message exceed the number of pixels available. Is important to make the comparison with the LSB in RGB method because in that one you could use 9 bits for one single position whilst in this case one pixel could be assigned to one pixel at most.	This method uses only Grey Scale images, in some cases the image used was RGB but it was transformed into GS using Matlab commands. This adds an immediate restriction because the positions available is the third part of RGB images. Nevertheless, it had the same problems exposed in LSB in Blue Layer because in that case the layer used behaves equally as the single layer of grey-scale images

Aspect	LSB in RGB	LSB in Blue Layer	LSB in GS
Message	The length of the message can have a size less or equal to the number of pixels. This means, using an image of 546x233, the length can be at most 127218 and that includes spaces as a character	Since the ASCII representation of characters has a maximum value of 255 using the extensions for special characters (the normal ones go until 127) and its binary representation is of length 8, the upper bound for the length of the message is $n/8$ where n is the number of pixels. This is using the worst case that each character has binary representation of length 8.	In this case, the length of the message has also an upper bound of $n/8$ where n is the number of pixels of the image. For example if the image has a size 46x16, the message must not exceed 92 characters including spaces.

Aspect	LSB in RGB	LSB in Blue Layer	LSB in GS
Stego-Image	Although this method was in the other aspects the best option, in the resulting image is the one that suffers more changes. Again, a pixel hides the information of a single character in the last three bits of each layer, meaning that the endings of the values can change from 000 to 111 and that increases (or decreases) the values 7 units. In the worst case all the three layers of the pixel increase or decrease by 7 units and that would be more visible for images with just a few pixels. An illustration of the situation can be found in figure 20, even if it is a subtle change it is more visible than increasing it just by one.	The problem that the stego-image might present with this method is that more pixels are changed and not only the ones that contain the information, the reason is because at one point the whole matrix turned its values into even values. If all the values were odds, the only pixels that will remain unchanged will be the ones storing a 1 inside them. However, this situation can also be seen as an advantage in the field of security, since an attacker with the original image might think that pixels that do not contain information do have information.	In the previously mentioned aspects, this method seemed to work much alike the LSB in GS but this aspect is probably the one that separates them. This algorithm not only just increases the values of the pixels in one unit, because in this particular case the changes are suffered just by one of the layers meaning that the colors will be changed less than in LSB in RGB and LSB in GS. That makes this implementation the best option in this aspect.

Figure 20: Changes in Stego-Image of LSB in RGB

Aspect	LSB in RGB	LSB in Blue Layer	LSB in GS
Algorithmic Complexity	Embedding and extraction are O(n), where n is the size of the message.	Embedding Algorithm is O(m x n), where m and n are the dimensions of the image. The Extraction Algorithm is O(n) where n is the length of the message.	Embedding Algorithm is O(m x n), where m and n are the dimensions of the image. The Extraction Algorithm is O(n) where n is the length of the message.

8 Steganography assisted by Cryptography

After studying steganography and cryptography it can be said that both of them are very effective techniques for keeping a conversation or a message covered, and may have the same objectives. However, as they have a lot of advantages they also have a lack of security in some way. Cryptography has a great level of security in terms of complexity of decryption, but if the aim of the emitter is to not raise suspicion about the message, here is where cryptography fails and this is when steganography comes into play. As well, steganography fails when the warden of the channel suspects something, in this case it will be easy for the warden to disembed the message, but if the text is encrypted then it will be so hard for the warden to read the message.

So after this analysis it was concluded that the combination of both techniques is a very effective way to keep a secret conversation between people. Hence, the algorithm to implement this combination was built, based on the ones that were already made. We used the LSB in RGB steganography technique, but the algorithm work for every steganography technique.

```
**LSB in RGB with cryptography: Encrypt and Embedding Algorithm**
Input image,message,X,Y,Key,n
Initialize: encryptedM ← $encrypt(message, Key, n)$
Initialize: stego-image ← $LSBinRGB(image, encM, X, Y)$
return stego-image
```

```
**LSB in RGB with cryptography: Extraction and Decrypt Algorithm**
Input image,X,Y,Key,n
Initialize: encryptedM ← $extRGB(image, X, Y)$
Initialize: decryptedM ← $decrypt(encryptedM, Key, n)$
print decryptedM
return decryptedM
```

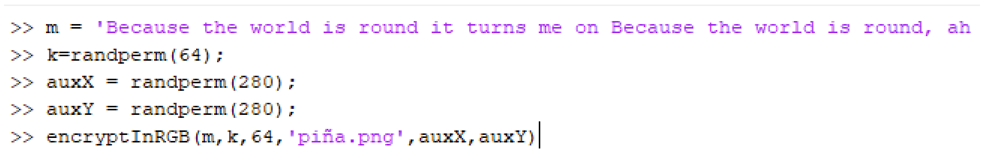
8.1 Example

To ilustrate the embeddig and extraction process assisted by criptography with a real implementation, the following text was used:

Because the world is round it turns me on
Because the world is round, ah
Because the wind is high it blows my mind

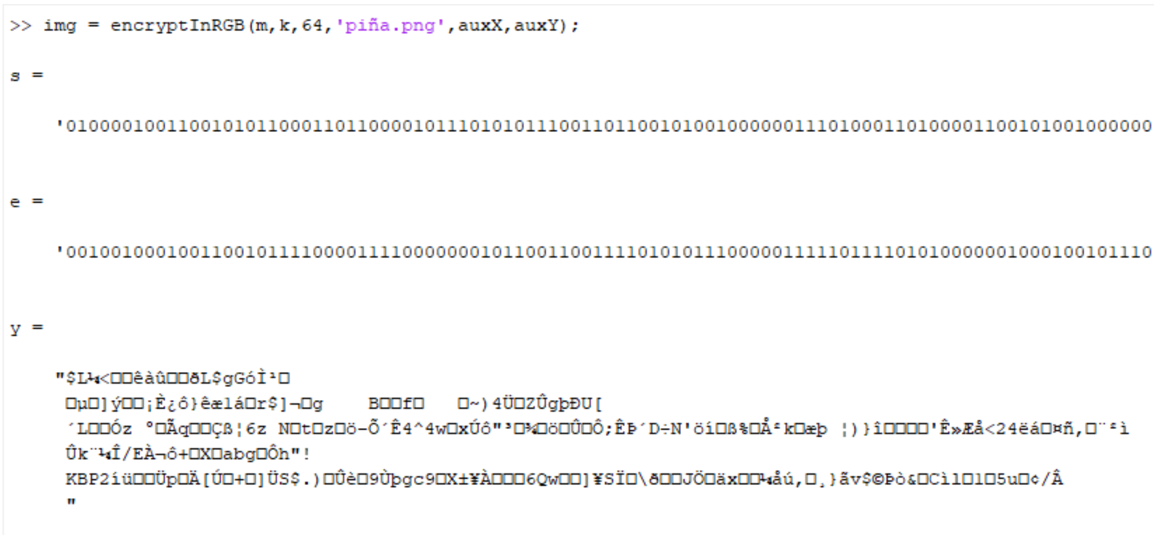
Because the wind is high, ah
Love is old, love is new
Love is all, love is you
Because the sky is blue, it makes me cry
Because the sky is blue, ah
Ah, ah, ah, ah
(The Beatles: "Because")

The algorithms were implemented in MATLAB, the fuctions names are; encryptInRGB for the one which encrypt the message an embed it in the image, and decryptInRGB for the one which extract the message from the image and decrypt it.



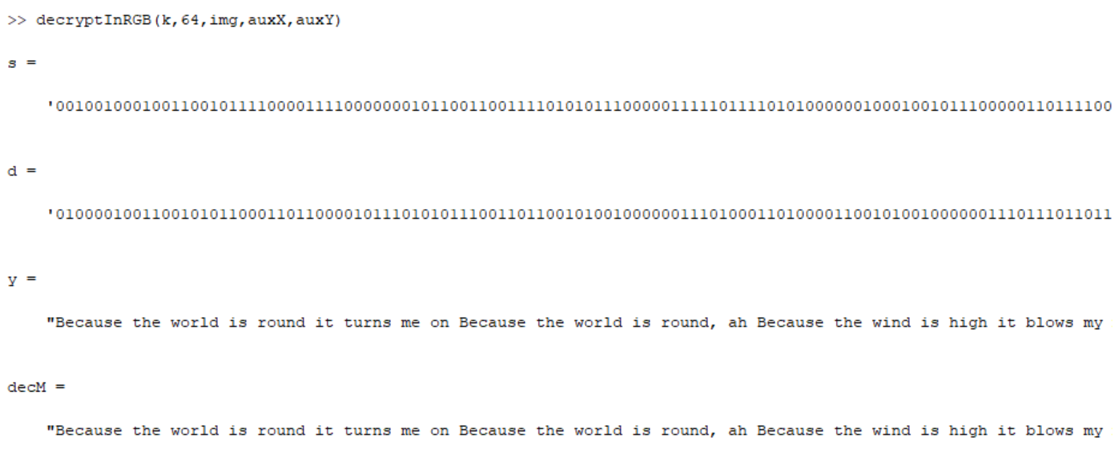
```
>> m = 'Because the world is round it turns me on Because the world is round, ah
>> k=randperm(64);
>> auxX = randperm(280);
>> auxY = randperm(280);
>> encryptInRGB(m,k,64,'piña.png',auxX,auxY)
```

Figure 21: Run of LSB in RGB with cryptography



```
>> img = encryptInRGB(m,k,64,'piña.png',auxX,auxY);

s =

    '0100001001100101011000110110000101110101011100110110010100100000011101000110100001100101001000000

e =

    '0010010001001100101111000011110000000101100110011110101011100000111110111101010000001000100101110

y =

    "$L¼<□□êàû□□ðL$gGóÌ¹□
     □µ□]ý□□¡È¿ô}êæ1á□r$]¬□g     B□□f□    □~)4Ü□ZÛgþÐU[
     ´L□□Óz °□Ãq□□Çß¦6z N□t□z□ö-Õ´Ê4^4w□xÚô"³□¾□ö□Û□Ô;ÊÞ´D÷N'öí□ß%□Å²k□æþ ¦))}î□□□□'Ê»Æå<24ëá□¤ñ,□¨²ì
     Ûk¨¼Î/EÀ¬ô+□X□abg□Ôh"!
     KBP2iü□□Üp□Ä[Ú□+□]ÜS$.)□Ûè□9Ùþgc9□X±¥À□□□6Qw□□]¥SÏ□\ð□□JÖ□äx□□¼åú,□¸}ãv$©Þò&□Cì1□1□5u□¢/Â
     "
```

Figure 22: Run of LSB in RGB with cryptography



```
>> decryptInRGB(k,64,img,auxX,auxY)

s =

    '00100100010011001011110000111100000001011001100111101010111000001111101111010100000010001001011100000110111100

d =

    '01000010011001010110001101100001011101010111001101100101001000000111010001101000011001010010000001110111011011

y =

    "Because the world is round it turns me on Because the world is round, ah Because the wind is high it blows my

decM =

    "Because the world is round it turns me on Because the world is round, ah Because the wind is high it blows my
```

Figure 23: Run of LSB in RGB with cryptography

After running the commands shown above (Figure 21, 22, 23) it can be seen in Figure 24 that the changes made on the original image are invisible to the eye, so it can be said the process was successful; and the extracted message is the same as the original.

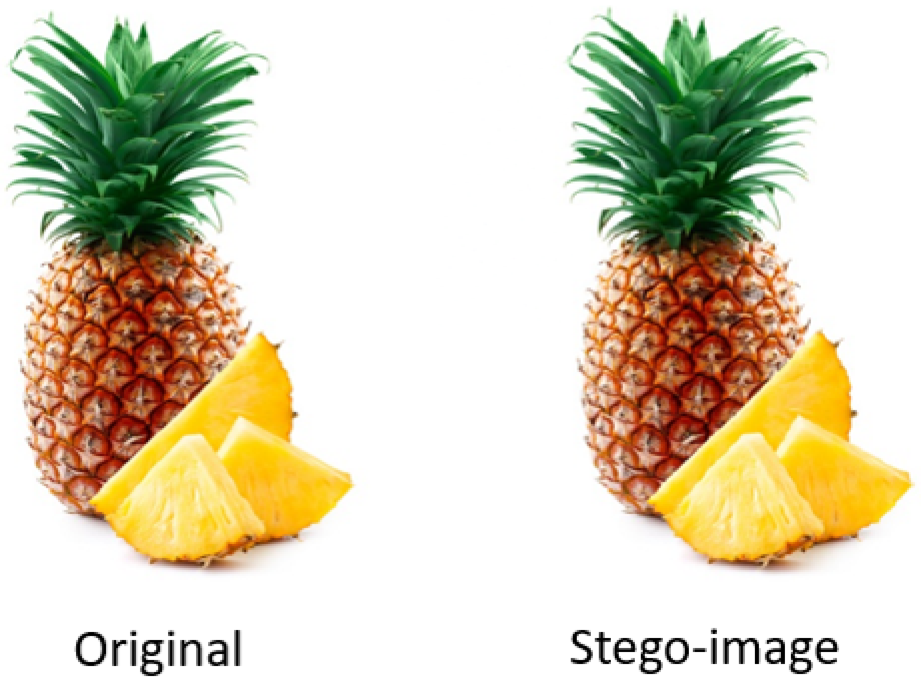


Figure 24: Changes in Stego-Image of LSB in RGB with a crypto message

References

[1] Ross J. Anderson and Fabien A.P. Petitcolas. On the limits of steganography. 1998.

[2] Johannes A. Buchmann. *Introduction to Cryptography.* Springer, 2000.

[3] Ingemar J. Cox, Matthew L. Miller, Jeffrey A. Bloom, Jessica Fridrich, and Ton Kalker. *Digital Watermarking and Steganography.* Morgan Kaufmann Publishers, 2008.

[4] Elke Franz, Anja Jerichow, Steffen Möller, Andreas Pfitzmann, and Ingo Stierand. *Computer based steganography: How it works and why therefore any restrictions on cryptography are nonsense, at best.* Lecture Notes in Computer Science, 2005.

[5] Jessica Fridrich. *Steganography in Digital Media: Principles, Algorithms, and Applications.* Cambridge University Press, 2010.

[6] Shilpa Gupta, Geeta Gujral, and Neha Aggarwal. *Enhanced Least Significant Bit algorithm For Image Steganography.* IJCEM International Journal of Computational Engineering Management, 2012.

[7] Mamta Juneja and Parvinder S. Sandhu. *An Improved LSB Based Steganography Technique for RGB Color Images.* International Journal of Computer and Communication Engineering, 2013.

[8] K. Muhammad, J. Ahmad, H. Farman, and Z. Jan. *A New Image Steganographic Technique using Pattern based Bits Shuffling and Magic LSB for Grayscale Images.* 2015.