

# LSB Digital Watermarking

---

I investigated the robust and imperceptibility of LSB encoding as a digital watermark<sup>1</sup>. I wrote a Matlab function to watermark an image. I wrote another Matlab function to decode the image, and a final function to break the encoding altogether by randomizing the LSBs.

## Algorithms:

There are many algorithms available for invisible digital watermarking<sup>2 3</sup>. The simplest algorithm is Least Significant Bit (LSB) Insertion,<sup>4</sup> in which each 8-bit pixel's least significant bit is overwritten with a bit from the watermark.

## LSB Encoding (and Breaking):

### Encoding:

I encoded my text string using 7-bit ASCII codes. I wrote a function, `encode.m`, to which takes as inputs the path to the image to use for encoding (`imagepath`), the text to encode (`text`), and the offset at which to begin encoding text (`key`). The function returns -1 on error; on a successful run it returns 0. After the function is run, it writes the array of bits it has modified (the `stenoimage`) to disk as `'stenoimage'`. This method works by taking a string of text, breaking each letter into its bit-wise representation and then overwriting the last bit in each pixel with one bit of the letter. For example, the lower case letter 'a' would overwrite 7 consecutive pixels' last bits with: 1, then 1, then 0, etc. for 0, 0, 0, and 1. In this manner we can encode an entire string as long as the string's letter to picture's pixel ratio is less than or equal to 1:7.

### Decoding:

The biggest downside to LSB encoding is that, even given zero information, an adversary can decode the contents of the encoding. Given the key (the first encoded pixel), the process of decoding is trivial. I simulate three scenarios: one in which the adversary is given the key, another in which the adversary does not have the key but has access to the original image, and a third in which the adversary has no information about the encoding.

The function `decode.m` takes as inputs: the method to use (as described above: 1, 2, or 3), the path to the original image (optional), the key (optional), and a guess as to the length of the key that is encoded

---

<sup>1</sup> [http://en.wikipedia.org/wiki/Digital\\_watermark](http://en.wikipedia.org/wiki/Digital_watermark)

<sup>2</sup> <http://www.cosy.sbg.ac.at/~pmeerw/Watermarking/>

<sup>3</sup> <http://www.cs.tut.fi/~selinum/watermarking/watermarking.pdf>

<sup>4</sup> <http://scien.stanford.edu/class/psych221/projects/06/itai/LSB.html>

(numletters). The function returns the method used on success and -1 on error. decode.m can be called in any of the following ways, depending on the information available (0 indicates null parameter):

Method 1: `decode(1,0,2718,45)`

Method 2: `decode(2,'images/lenna.512',0,45)`

Method 3: `decode()`

The methods are covered in detail below:

### ***Method 1:***

This is the simplest method. Given the key and a length, this method will print the encoded text. It works by gathering each pixel's last bit, starting at the key value and continuing for length times 7 pixels.

### ***Method 2:***

Given the original image and a length, this method compares the 2 images to find the key and then the message. The method works by XORing the pixels of each image against each other, one by one, until it finds pixels that differ.

### ***Method 3 (default):***

Given no information, this method will scan the LSBs for any ASCII patterns, one offset at a time, rating each string as it finds it, keeping track of the most English-like string. The method then returns two guesses: the best ratio of letters to non-alphabetic characters and the best word-length. The user is then given the word length value for the best ratio and the ratio for the best word length. It is up to the user to determine the key, given this information.

Method 3 relies heavily on `identifystring.m`, which is used to identify how English-like a string is. The `identifystring` function determines the percentage of characters in a string that are in the alphabetic ASCII range. It returns the ratio of letters to total characters by default but will return the ratio and average letters if there are two output arguments.

### ***Breaking LSB Encoding:***

LSB encoding can be trivially broken by randomizing the last bits in an image. Only by comparing the broken image to the original image can one tell that it has been tampered with. The function `breaking.m` takes an image to break as the input, returns -1 on error and 0 on success, and writes the broken image to disk.