Amelia Rave, Danielle Phillips

Steganography: Hiding Information in Media Files

1. Introduction

In cryptography, a lot of time and effort is spent figuring out how to make schemes secure against adversaries who are monitoring networks or eavesdropping. Cryptography tries to make a message impossible to read unless you are meant to be reading it. Steganography, on the other hand, aims to hide the very fact that a message was sent from adversarial monitors. The word steganography comes from the Greek word meaning “covered writing”, whereas cryptography means “secret writing”. The two areas of study have the same goal of keeping information out of the hands of an adversary but approach the problem differently.

Primitive forms of steganography have been practiced throughout history. A Greek ruler named Histaieus wanted to tell his allies that it was time to start a revolt against the Medes and the Persians, so he shaved the head of one of his slaves and tattooed the message on his head. Histaieus waited for the hair to grow back, and then sent the man along to his allies. This worked as a way of concealing a secret message although the message itself was not a code. All Histaieus had to do was make sure the intended receivers knew where to find the message, thus the revolt succeeded.

In part, developments in modern steganography have been motivated by the flaws of cryptography such as the lack of strength and governmental intervention that limits or prohibits certain cryptographic schemes. These weaknesses have prompted people and businesses alike to search for alternative ways of communicating sensitive information. Furthermore, steganography’s rise in popularity is also due to the increased tech-savviness of the general population. It stands to reason that hiding information in an unsuspecting image attracts much less adversarial attention than sending a clearly encrypted file.

2. Steganography and Images

Steganography today works by hiding information inside of other information such that the original data appears as normal. Almost any digital file format such as text, image, audio, video, and protocol can be used for steganography. However, digital images are the most widely-used vessels because of how plentiful they are on the Internet. Moreover, a digital file format that bears a lot of redundancy is more desirable for steganographic use, because it is easier to hide information without detection. These kinds of digital file formats include bits that provide accuracy exceeding those necessary to display and use the file. Thus, there are plenty of practically unused bits that can be utilized by steganographic techniques to embed data. This is the reason why digital file formats with redundancy are the preferable kind for steganography—it is easier to embed data without detection when there are unused bits included in the file. To hide secret data in an image, there are multiple techniques available all with their own strengths and weaknesses. Since there are multiple different image file formats like png, jpeg, gif, tiff, psd, pdf, eps, and ai, there are also different steganographic algorithms for each because they perform image compression differently.

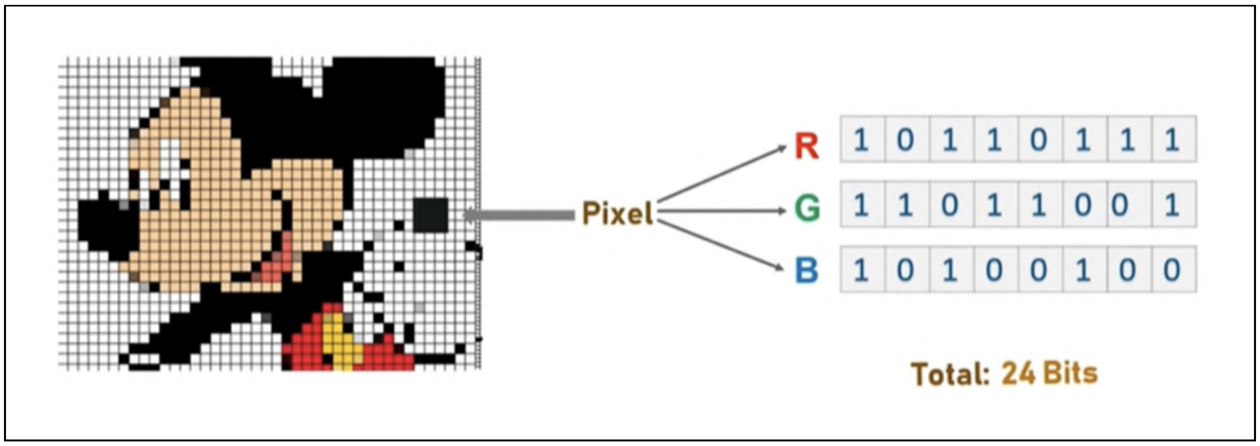
Image compression is done because the actual size of many image files is too large to transmit over a standard Internet connection, so the image is compressed to a smaller size while still displaying the intended picture. An image is a collection of numbers, represented in binary, that represent different colors or light intensities in different locations. This idea is implemented as pixels that form a grid where each pixel has a location and coloring in the grid that contribute to the overall image. The number of bits used for each pixel, or the bit depth, dictates the number of different colors that can be displayed by a single pixel. For example, the least amount of bits currently used for a single pixel is eight. This means that there are eight bits used to describe the light intensity, or color, of a single pixel. With a bit depth of eight, we have a monochrome or grayscale image where each pixel can display 256 different shades of one color. Usually, digital color images use a bit depth of 24 where each of the primary colors red, green, and blue is represented by 8 bits. Here, there are 256 different intensities of red available, 256 intensities of green, and 256 intensities of blue, summating to over 16 million combinations of colors. Clearly the human eye will not be able to differentiate between so many different shades of the same color, and the more colors that can be displayed the larger the file will be, thus there is an opportunity to remove unnecessary detail and reduce file size. This is the reasoning that leads us to explore the concept of image compression as it relates to steganography with image file formats.

As stated before, images files are often too large to be reasonably transmitted over standard Internet connections, thus file size must be reduced while preserving the integrity of the image. The process of analyzing and condensing image data to reduce file size is called image compression. There are two types of image compression called “lossy” and “lossless”. Lossy compression discards image details that are too minute to be noticed by the human eye, so resulting images are not an exact duplicate of the original image. Joint Photographic Experts Group (JPEG) is an example of an image file format that uses lossy compression. Alternatively, lossless compression does not remove any information from the original image, but represents data in mathematical form to reduce file size. Using lossless compression, a smaller image file is identical to the original image. Graphical Interchange Format (GIF) is a popular image format that uses lossless compression. The way an image file is compressed dictates which steganographic algorithm should be used for both hiding and extracting messages from an image. Using lossy compression, an embedded message may be partially lost when removing excess image data, so a specific steganographic algorithm must be used to address this case.

Techniques for image steganography are divided into two groups: Image Domain and Transform Domain. Image domain techniques embed messages into the intensity of pixels directly, while transform domain techniques transform the image first and then embed the message. Image domain techniques are best used with image formats that are lossless, since the message is embedded in the pixels directly, we do not want lossy compression to remove parts of the message. Steganography in the transform domain manipulates algorithms and image transformations, meaning messages can be hidden in more significant areas of the cover image. This makes transform domain steganography more vigorous and powerful than image domain steganography, the latter sometimes being referred to as “simple”. Furthermore, transform domain steganography does not depend on the image file format, so the hidden message can survive conversion between lossy and lossless compression. This is a significant benefit of transform domain over image domain, because no matter how the image is transferred or compressed, the embedded message persists. Different steganographic algorithms are categorized by the image file formats used and whether the algorithm is performed in the image domain or transform domain. Now we will look toward the process of embedding data in digital image files using a common steganography algorithm.

Embedding Data in Images

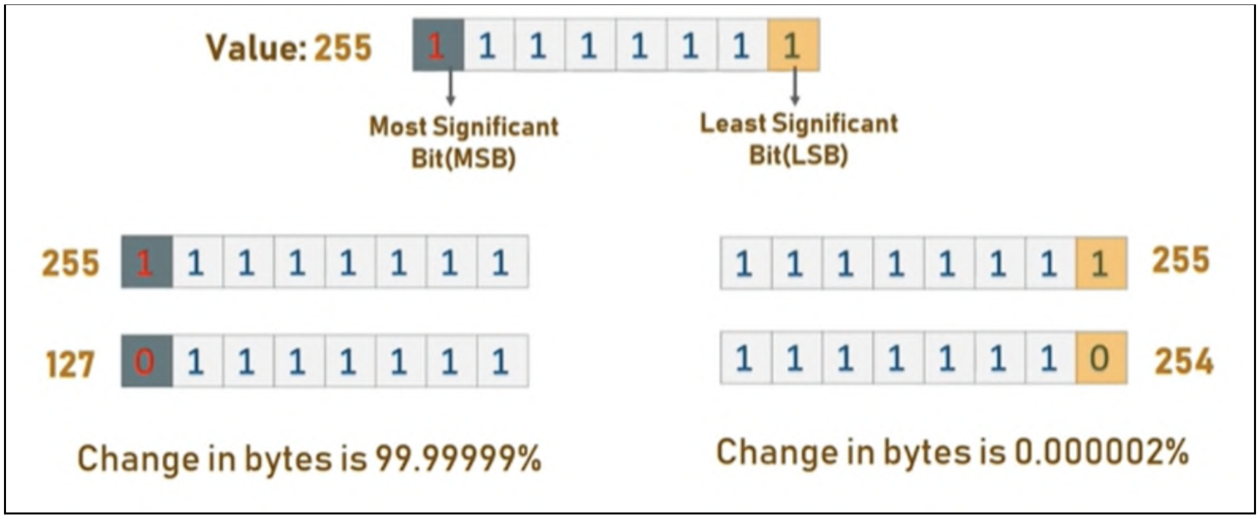
Least significant bit (LSB) insertion is a simple and common steganographic algorithm for embedding information in an image and falls within the image domain explained above. In basic LSB steganography, some set number (specifically, the length of the message encoded) of each pixel's least significant bits in an image are replaced with some portion of a message. Essentially, every image that has colors is made up of pixels with 3 attributes to describe the color. We will be representing these values using the RGB method, which uses the values red (R), green (G), and blue (B) to describe the color of each pixel in the image. This is just for simplicity; there are other 3-dimensional ways to represent pixels, such as HSV (hue-saturation-value), but we will not be focusing on those for the duration of this paper. Combinations of these three colors of light can represent any color on the spectrum of visible light, although this is limited in digital images by how much space is allocated for color values. The standard is for each color to be represented by 8 bits, meaning each color has a value between 0 and 255. If all colors are at the value 0, the pixel will be black since there is no light value, and if all the colors are at value 255, the pixel will be white. The figure below offers a visualization of this kind of representation of pixels.



Credit: https://www.edureka.co/blog/steganography-tutorial

It may seem as though changing the values of the colors in each pixel might drastically change the outcome of the image overall. How do we ensure the preservation of the original image?

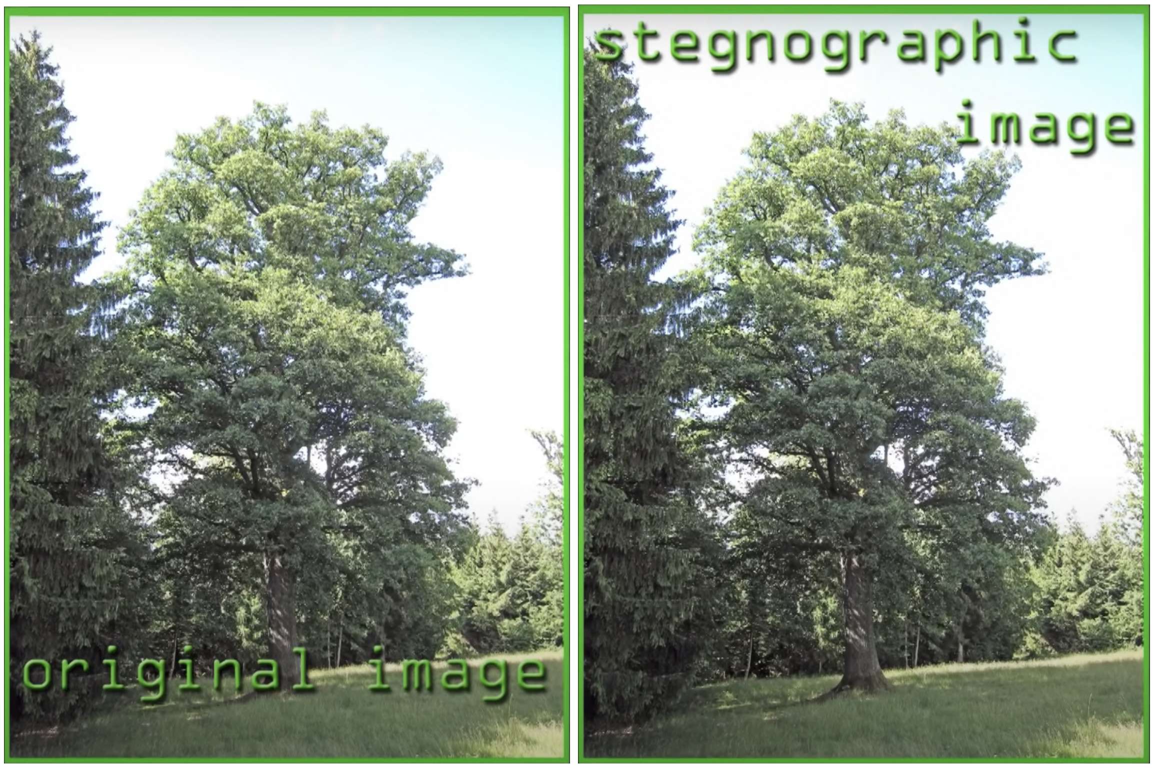
This is the reasoning behind the use of the least significant bits of each pixel. It has been found that changing the smaller binary values does not have a very noticeable effect on the overall image. If someone were to encode the most significant bits of the color values, it would have a drastic effect on the overall image. The diagram below demonstrates why this is.


Value: 255
1 1 1 1 1 1 1 1
Most Significant Bit(MSB)
Least Significant Bit(LSB)
255 1 1 1 1 1 1 1 1
127 0 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 255
1 1 1 1 1 1 1 0 254
Change in bytes is 99.99999%
Change in bytes is 0.000002%

Credit: https://www.edureka.co/blog/steganography-tutorial

The sender decides how many bits they would like to encode. For example, if the sender chooses to use the 2 least significant bits, they would be choosing to change the last 2 bits of every pixel's color values in order (for example, R, then G, then B) to contain two bits of the message. This would mean that every pixel would contain 6 bits of information. This means that a single pixel has $2^6 = 64$ possible values that can be encoded. If the sender wished to send the binary value of the message using traditional ASCII characters, which have 128 possible characters, this would mean that a single letter could be stored every 2 pixels. This may seem like it would take up a lot of space, but pictures today have become very high-resolution and typically hold tens of thousands of pixels. Splitting this number in half to allow one two pixels per character still allows thousands of characters to be encoded in the image.

If the sender chooses to use less bits, the image will have a less noticeable difference, but the message would have to be shorter to account for space lost. On the contrary, if the sender chooses to use more bits, there will be more space to hide longer messages, but they now run a higher risk of an adversary noticing that the image has been altered. To demonstrate how little this visually changes the image, below is an example of an image that has been changed using standard 2-bit LSB to hide the binary form of the entire works of Shakespeare.



Credit: https://www.youtube.com/watch?v=TWEXCYQKyDc&ab_channel=Computerphile

Clearly, this method can hide large messages within images with seemingly no difference. Without the help of a computer analyzing these images, it is virtually impossible to tell that anything has been changed. However, using only this method on the first few pixels makes it reasonably trivial for adversaries to determine that the image has been altered just by looking at the steganographic image. We will discuss the reasons for this later.

Alternatively, the sender can choose to alter the DCT (Discrete cosine transform) coefficients of the image. This is an example of a Transform Domain technique. This uses LSB but determines which pixels are changed using a mathematical algorithm. Basically, an image can be converted into a series of cosine waves whose unique combination of coefficients define the image. Changing these coefficients instead of the direct pixels will make the secret message much harder to detect, as the added noise to the image will appear much more random. However, this method alone, similar to LSB, makes it relatively easy for an attacker to recognize that there is a message encoded but makes it much harder for an attacker to decode that message than LSB. When it was first implemented, researchers found that it was possible to track the frequencies of the coefficients within different blocks of the image and compare them to what they should normally be. Again, we will discuss this in more depth later, but as a result, modern implementations of DCT steganography will track these distributions of values of coefficients and try to balance them throughout the image in order to defend from these kinds of attacks. This makes it much harder for an attacker to recognize that there is anything encrypted in the message.

Extracting Data in Images

In the case of basic 2-bit LSB steganography, the receiver would extract the last 2 bits of every pixel and concatenate them all to reveal the intended message. In the case of DCT steganography, the sender would need a decoding algorithm that transforms the image into its series of cosine waves to extract bits from those coefficients. Once they complete that, just like standard LSB, they would concatenate all the modified bits into a readable message. Either way, the receiver would need to know beforehand how to extract the message; just as with every other form of encryption, the receiver must know something about the encryption that other adversaries don't.

What is different about decoding in steganography versus cryptography is that it is not assumed that the adversary knows the algorithm to decode the message in the image; rather, it is assumed that the adversary only knows that an image was sent. In cryptography, the adversary generally knows the algorithm that was used to create the encrypted message but is lacking a key that the intended receiver has. This is because in steganography, as stated before, the image sent is supposed to hide from everybody that it is even encrypted at all. This contrasts the purpose of cryptography, which is supposed to simply make messages unreadable by adversaries, even though they can see the message that is sent and the algorithm used to create that message. This is why the receiver typically just has a decoding algorithm that can be used to extract the message as opposed to a key that applies to an open and public decoding algorithm.

4. Attacks on a Steganographic System

An attack on a steganographic system is successful if an adversary is able to detect that there is information hidden inside a file. What does such an attack look like in practice?

We will preface this section with a disclaimer that the best-case scenario of sending a message in an image (the scenario where nobody suspects that an image was tampered with) requires that the original image is never released to the adversary. If the adversary has the original image, they can perform a subtraction between the images to see if anything has been altered, and if it has, they have basically already succeeded in detecting that there is information in the file. For that reason, it is best in practice to send images that were taken by the sender as opposed to images that were found online or are owned by anybody else, because the sender can then delete the original image after it has been edited, ensuring that nobody will ever see the original image. For the rest of this section, we will assume that the adversary does not have access to the original image and must therefore use only the clues within the steganographic image to determine whether it holds a secret message.

There are multiple kinds of attacks on steganographic systems. An attack someone might use to detect very simple steganographic encryptions is called a visual attack. This is where an attacker would isolate the least significant bits of every pixel and represent them visually. As it turns out, even though the least significant bits of a pixel do not heavily affect the image, they still follow the general trend of the image. Pixels that are darker tend to have more least significant bits of value 0, whereas pixels that are lighter will tend to have more pixels that have value 1. As a result, the least significant bits of an image should vaguely resemble the overall image. However, when these bits are altered, they will no longer resemble the image when isolated. Below are images of the least significant bits represented in grayscale of a picture of a windmill next to the least significant bits of the same image that has been altered using standard 2-bit LSB alterations.

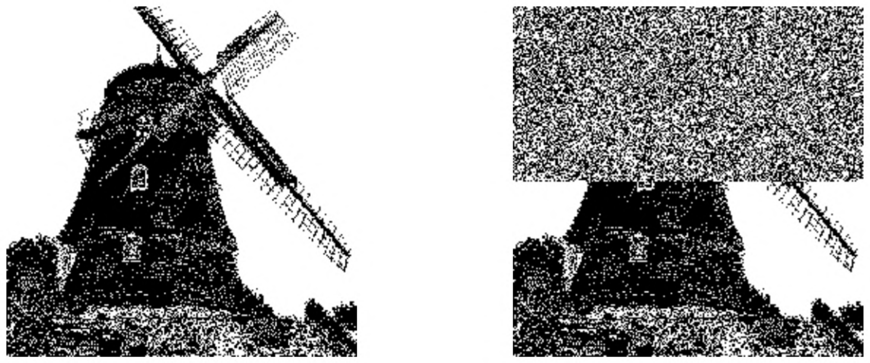
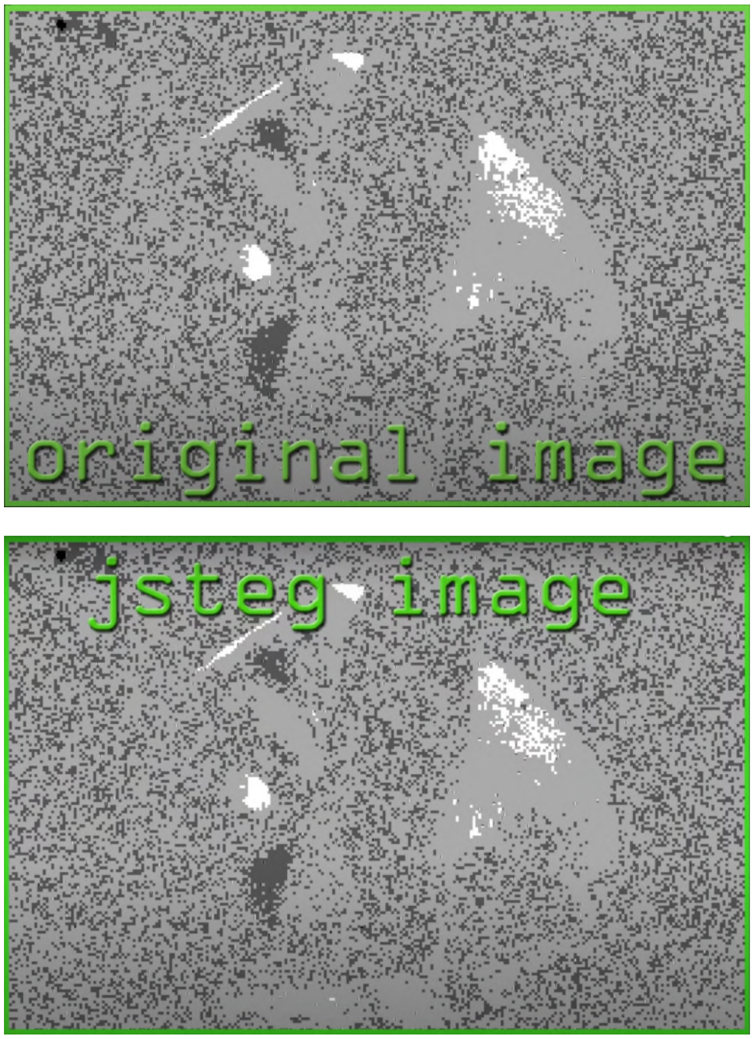


Fig. 7. EzStego; filtered images of Fig. 4: nothing embedded (l.), 50 % capacity of the carrier used for embedding (r.)

Credit: *Attacks on Steganographic Systems*, A. Westfield and A. Pfitzmann

This shows how easy it is for an attacker to recognize when a traditional LSB method has been used to alter the picture. If the message is not long enough to cover the whole image, it is very easy to see where the message ends. If the embedded message is not encrypted, the attacker could easily concatenate the bits together and reveal the message. This demonstrates why it may be beneficial to encrypt a message that is steganographically embedded in media if the message is very important or holds very sensitive information that cannot be seen by others.

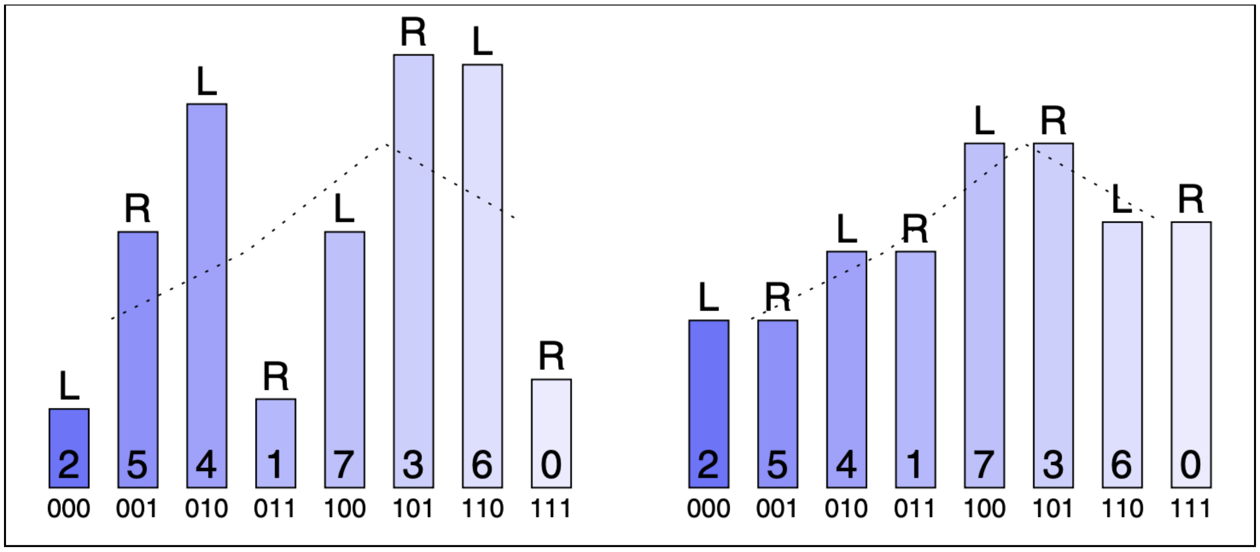
As a result of visual attacks, as stated before, modern steganographic algorithms will try to mirror the distribution of the pixels and mirror generally what they should look like. Softwares and libraries such as JSTEG do a very good job of this. It uses DCT and analyzes which bits will create the least noise when swapped with specific parts of the message. Trying to preserve the proportional distributions of the original image's least significant bits will prevent an attacker from noticing noisy bits during least significant bit isolation. Below are the least significant bits of a picture of a panda (top) next to the least significant bits of the same image that has undergone a JSTEG transformation (bottom).



Credit: https://www.youtube.com/watch?v=TWEXCYQKyDc&ab_channel=Computerphile

As you can see, the least significant bits of the steganographic image on the bottom are very similar to those in the original image at the top. Upon close investigation, it is noticeable that the pixels are slightly different, but from the perspective of the attacker who does not have access to the original image, the picture seems to have a similar distribution to the original image. At this point, a human could not use a visual attack to determine that there is a message embedded within an image; they would need the help of a computer to notice patterns.

This brings us to the concept of statistical attacks. These are done by computers who can analyze the exact pixels and determine whether they have a normal distribution among the image or not. A common form of this attack is the chi-squared attack. Its exact mathematical calculations are out of the scope of this class, but basically, it uses the theoretical expected values of frequencies throughout the image and compares them to what is portrayed in the image. A visualization is shown below to describe how it compares pixel frequencies.



Credit: *Attacks on Steganographic Systems*, A. Westfield and A. Pfitzmann

An algorithm like this might have a better chance of discovering the fact that there is a hidden message in JSTEG embeddings, like that of the panda image shown above. Avoiding automated statistical attacks is the main focus of embedding messages today, since it is very hard to come up with an algorithm that is guaranteed to fly under the radar when under close statistical scrutiny.

5. Watermarking and Fingerprinting

The rapid development of digital technology and widespread use of the Internet to share media has created a need for the ability to demonstrate ownership of intellectual property. A huge number of digital images and videos are shared and generated daily by businesses and civilian users alike. As a result, security and copyright management is faced with new challenges. In response to these challenges, the techniques of watermarking and fingerprinting digital media have emerged. Watermarking and fingerprinting are closely related to steganography but have slightly different motivations. Fingerprinting is the practice of digitally marking an object so that intellectual property may be claimed and owned. Watermarking, on the other hand, usually entails embedding unique marks in distinct copies of an object so that the owner can identify the specific customer who illegally shared the object beyond the scope of an agreement. In both watermarking and fingerprinting, the fact that information is embedded in a digital media object need not be hidden to serve its purpose, unlike steganography where this fact must be concealed. In the case of watermarking and fingerprinting, an attacker aims to remove the embedded data or mark from the digital file, thus stealing the intellectual property.

6. Conclusion

Steganography has come a long way since its origins in ancient greece. Many new steganographic algorithms have been developed over the years and continue to emerge, catering to more digital file formats than ever before. Steganography allows for the concealed transfer of private messages and is also commonly used today in copyright and intellectual ownership cases by utilizing watermarking and fingerprinting. Steganography gives users a way of communicating secretly without dealing with the pitfalls of cryptography nor the elevated mathematical complexity of using a secure cryptographic scheme. This paper went into detail about the commonly used least significant bit image steganography algorithm, however we acknowledge that there are many more algorithms and methods of embedding messages in digital media files. With the multiple methods of embedding information into different types of media, we wonder how steganographic advances may apply to new media file types in the future. Furthermore, we look forward to keeping an eye on new steganography analysis tools as they become more robust and capable of detecting multiple kinds of steganography techniques.

Sources

Chang C-C, Lin C-C, Yang C-N. Steganography and Watermarking. Nova Science Publishers, Inc; 2013. Accessed April 1, 2022. https://search-ebscohost-com.proxy.lib.umich.edu/login.aspx?direct=true&db=e000xna&AN=591935&site=ehost-live&scope=site

Choudary, Archana. "Steganography Tutorial - A Complete Guide for Beginners". https://www.edureka.co/blog/steganography-tutorial

Computerphile, *Secrets Hidden in Images (Steganography)*. https://www.youtube.com/watch?v=TWEXCYQKyDc&ab_channel=Computerphile

Khan, David. The History of Steganography (https://doi.org/10.1007/3-540-61996-8_27).

Roy, Rupali. "Image Steganography Using Python". https://towardsdatascience.com/hiding-data-in-an-image-image-steganography-using-python-e491b68b1372

T. Morkel, J.H.P. Eloff, M.S. Olivier. "An Overview of Image Steganography" https://www.academia.edu/2943481/An_overview_of_image_steganography?sm=b

Video portion of the project link:
https://drive.google.com/file/d/1oQHvrKp3HTmGVjCH_RAON1vjVKtMRvYo/view?usp=sharing