CS 198 Codebreaking
Fall 2022 Ryan Cottone, Imran Khaliq-Baporia **Note 8**

Steganography

Even in theoretical cryptography, there are situations in which we don't have a reliable channel to send information through. Vigilant authorities might block the transmission of any messages that appear to have a secret message, or are encrypted. Adversaries might even have a secret method to break our encryption, so brazenly sending encrypted messages would be insecure. Normal circumstances usually do not call for such heavy constraints, but there are specific situations in which it is paramount to communicate regardless of these adversaries.

Before we consider how encryption might help us here, we might want to consider a different approach entirely. Other methods of keeping data secret do exist, most of which attempt to 'encode' data where attackers are unlikely to look without knowing the secret. These methods are often called **steganography**.

In fiction, these would often appear as a message hidden in a letter, often as the first letter in each sentence comprising the entire message. [1] Some spies even would hide information in crossword puzzles!

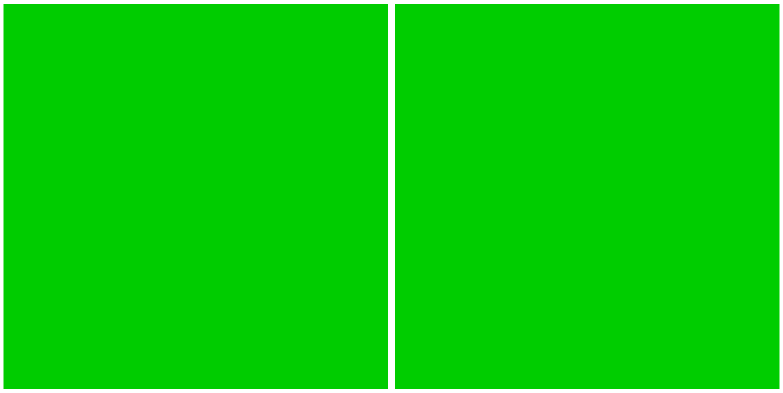
Often times, however, we want to use more modern methods of transmitting secret messages using steganography. Still, we need to keep up the appearance of a totally normal message, which can be tricky when modfiying an image. Key to this plan is the nature of computer-based graphics, which provide a large amount of flexibility for encoding information. If we only modify the last bit of the bit representation of a pixel, the overall image appears unchanged.

Digital Images

To explore how this is possible, we must first look at how images are stored within computers. At the most basic level, we need to represent each pixel as a combination of red, green, and blue. Computers do this by assigning each color a value from 0 to 255, representing (roughly) the intensity of that color within the pixel. For example, (255, 0, 0) is all red, (0, 0, 0) is white, and (255, 255, 255) is black.

As a result, we can represent a wide range of colors as a combination of these three (a total of $2^{24} = 256^3 = 16777216$). Note that each color is represented by 8 bits ($2^8 = 256$), and there a binary string. The nature of binary makes it so modifications to the least significant bit only change the overall number by at most 1. The change in color intensity is nearly impossible to detect by the naked eye: take a look at the following (very large) pixels:

[1] If you haven't found the hidden message yet, I recommend reading this line again!

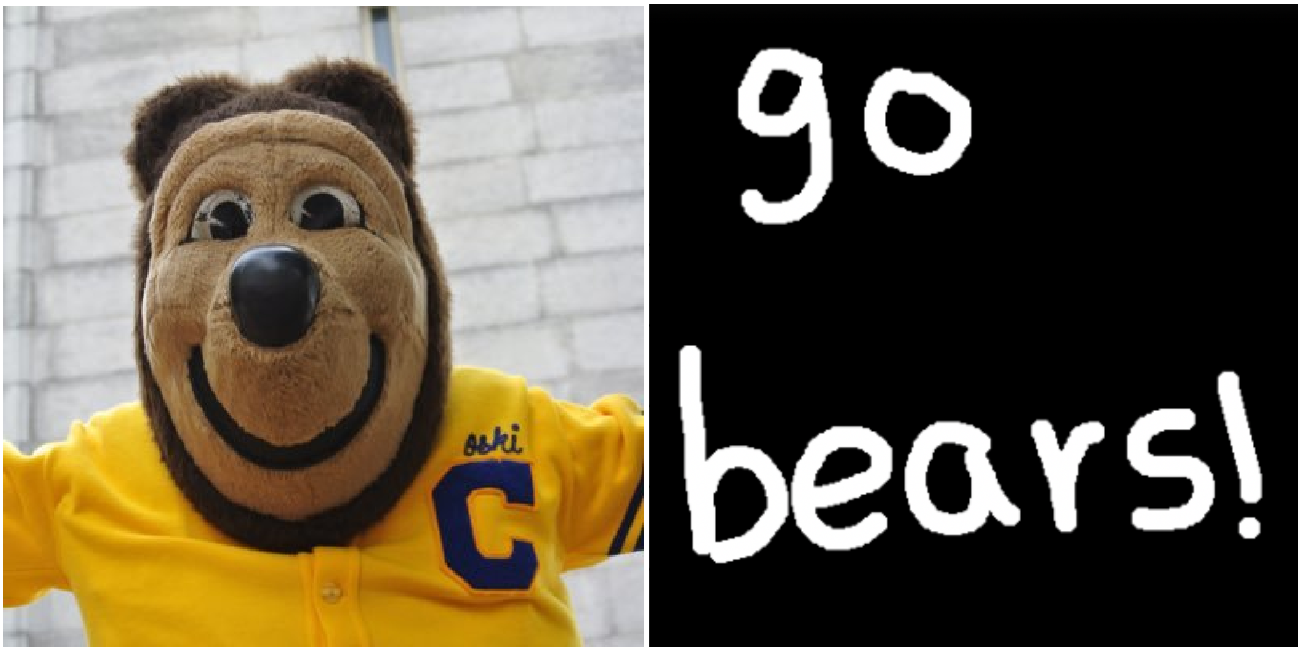


One is [0,204,0] and the other is [0,205,0]. Humans simply cannot tell the difference at a glance (especially when its one tiny pixel alongside millions of others).

Encoding Information in Images

In order to encode a message, we then need a scheme for changing the values of the last bit to something specific. We could always just denote a normal bitstring message as the result of concatenation every last bit of each pixel. This would work for text-based messages.

Image-based messages can be done in a similar fashion. If the last pixel of the image is 0, we set the message pixel to white, and black otherwise. This forms a new, black-and-white image that can convey a message.



The image of Oski on the left encodes the image of "go bears!" on the right.

Crucially, however, images provide a far better **expansion ratio**, or how much more "normal message" is needed to hide the secret message. For steganography like encoding a message as the first letter of every sentence, the expansion ration is abysmal – probably 1 character of secret message to every 50 of normal message. For images, each pixel can transmit multiple bits of information. In a 1024x1024 image, that's hundreds of kilobytes of data!

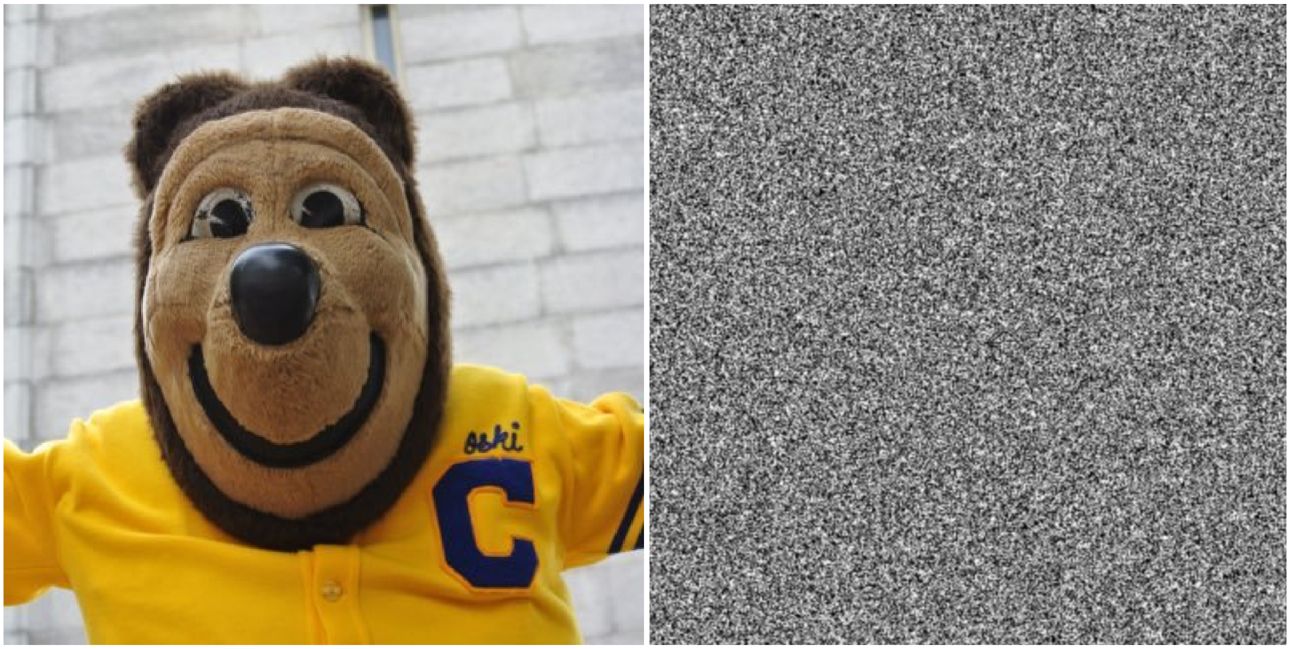
Steganalysis

Methods do exist to try and detect when steganography has occured, but it is difficult to do so without knowing where to look. The situation is also highly dependent on whether the stegoanalyst has access to more than one stegographically-modified images, or if they even know steganography has occured at all. Overall, steganalysis is considered incredibly difficult.

In certain cases, it is possible to compare the expected proportions of data to the observed proportions. If the last bits of a large image were made of 80% 0s, something is probably fishy – it is exceedingly unlikely for that to happen by chance except in rare circumstances (like a lot of whitespace in the background).

Content Threat Removal

While it may not be easy to detect whether a message is steganography, it is easy to mess with the outcome. Since we have seen that modifying the last bits of an image has negigible effect on the overall image, if we randomly scramble them, it should have no real effect either. Randomly scrambling the last bits of every image would also ruin any messages send by steganography, so there is little downside to doing so. Of course, this is dependent on knowing the scheme used for steganography.



This the result of running our decoding algorithm on a scrambled image, while the original still looks fine.

Contributors:

- Ryan Cottone