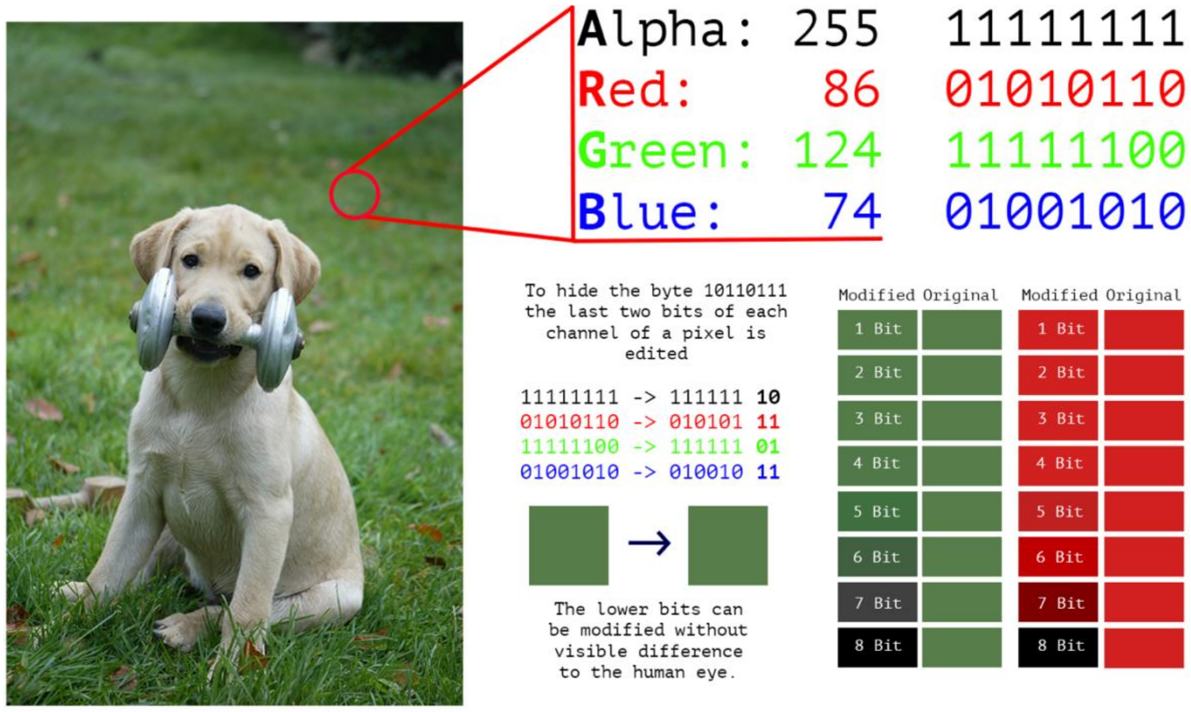
Q1:

I chose images because my understanding of image encoding is better than my understanding of audio encoding. I chose to use PNG files for my stenography project for a few reasons. I wanted to use either PNG or JPEG since those are in my experience the two most common file types for images. JPEG uses a lossy compression algorithm; this means that after I modify some pixel values the content of the image file may still change causing corruption to the data hidden in the image. To my understanding this can be worked around by changing the content of the already compressed file. However instead I just chose PNG since the lossless compression means I can modify pixels exactly to my liking without risking any corruption to the data I store in the image.

Q2:

Some metadata is stored in the file. The first one is the length of the expected data and the second one is a signature that there is data to find and lastly the depth which is the number of bits changed in each byte. This is done by changing the least significant bit of the of each channel of first 8 pixels, in total 64 bits. Of these 64 bits the first 32 is the *signature*, 32 bits that need to match a hardcoded value. This is so that the decoding knows if there is data to look for. The next 3 bits is the *depth,* this is the number of bits of each byte that is changed when the data is stored in the image. 1 bit per byte is hardly visible while 8 means that the image is changed entirely. Lastly the next 29 bits encode the size of the data in the image. This means that the maximum data that can be stored in a single image is 536 870 912 bytes or about half a terabyte which is well beyond the size of most images. Thereafter the data begins, meaning that the metadata consists of a total of 8 bytes.

Q3:



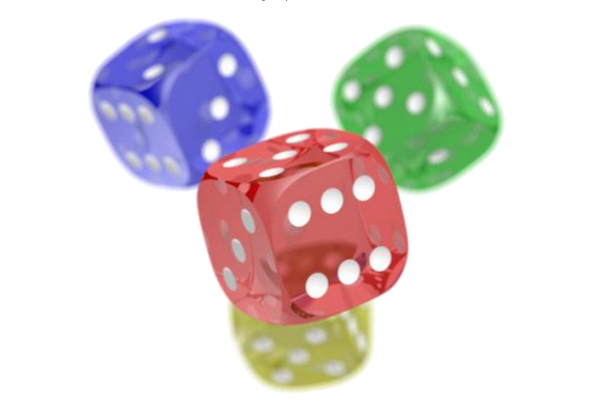
To hide the byte 10110111 the last two bits of each channel of a pixel is edited

```
11111111 -> 111111 10
01010110 -> 010101 11
11111100 -> 111111 01
01001010 -> 010010 11
```

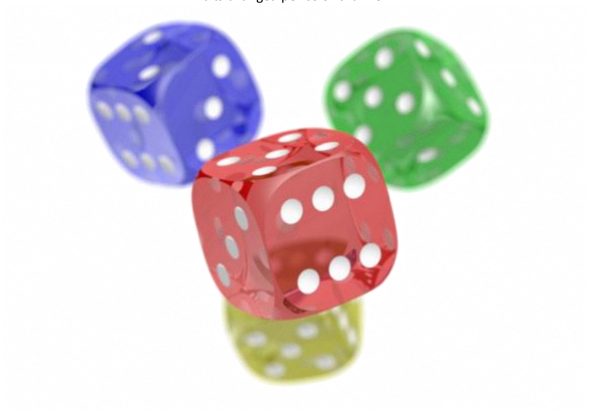
The lower bits can be modified without visible difference to the human eye.

In my testing the 2 least significant bits could be changed without noticeable difference. At 3 bits changed the image would still look good but it would be visibly worse than the original image. On the following page the same has random data hidden in it with different number of bits per byte changed.

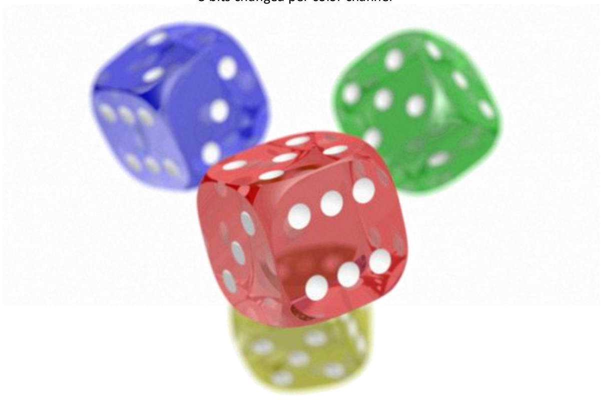
1 bit changed per color channel



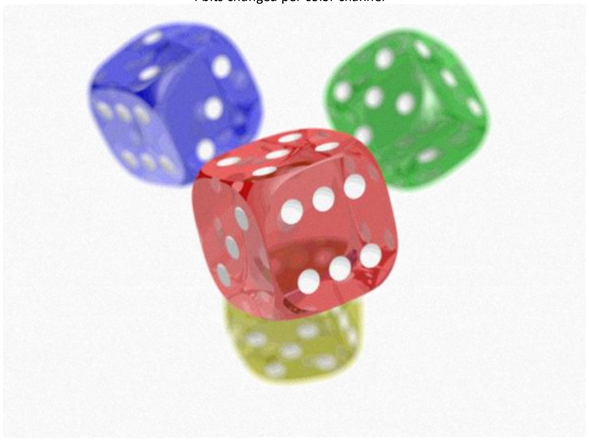
2 bits changed per color channel



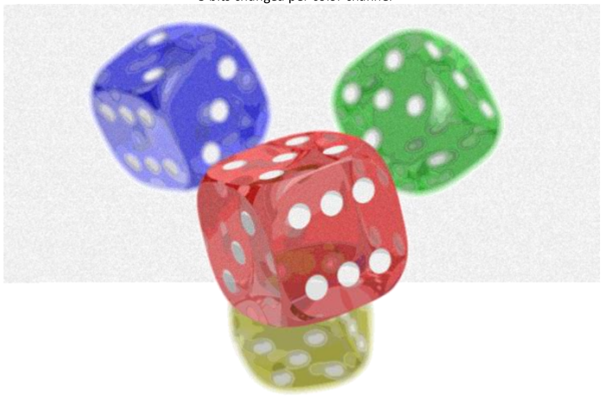
3 bits changed per color channel



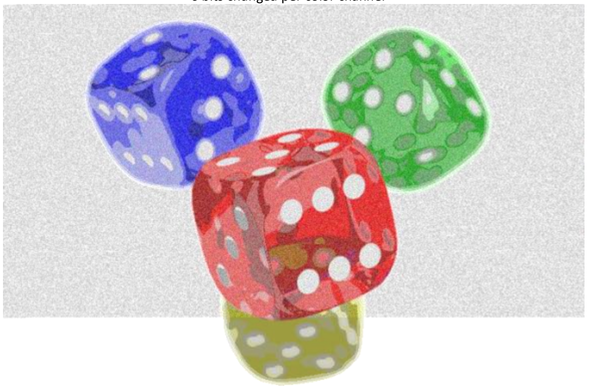
4 bits changed per color channel



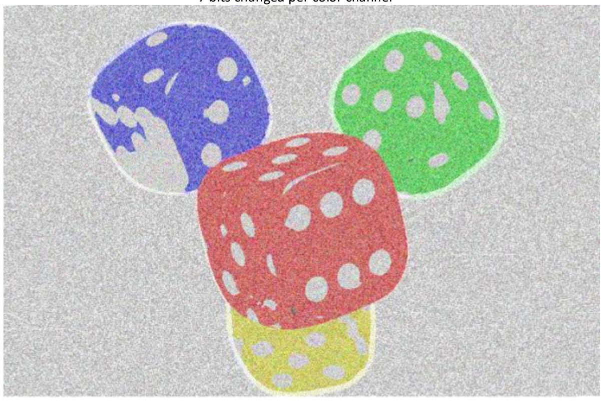
5 bits changed per color channel



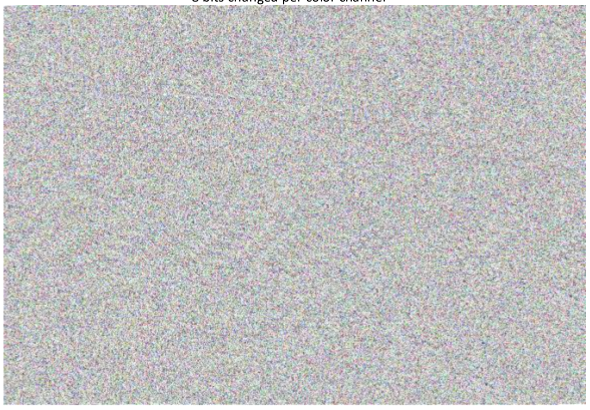
6 bits changed per color channel



7 bits changed per color channel



8 bits changed per color channel



Q4:

When I chose the value of c’ I simply set the last n bits of the color channel byte to the value of the data I want to hide in the image. This technique works best on photographs and images that naturally have a lot of noise and small changes in pixel values are expected while images that use a lot of the same color will suddenly have some variation in the least significant bits which is a clue that the image has been altered and that it may contain a message.