Intro to Security
Assignment 1 Report
Gregory Newman

Task 1

I executed this portion of the assignment on my Linux machine using the openssl commands in the terminal. I generated encrypted pictures of both the Beaver logo and a personal picture of myself using both the CBC and ECB encryption modes. For the Beaver logo, the ECB encryption left a very obvious impression when viewed with image software. That is because there is a lot of repeated data in the file (the picture is only made up of 3 colors) and it is all encrypted using the same key. The CBC mode encryption did not produce an impression of the image because additional randomness is added to each block through the use of an initialization vector, with the ciphertext of the previous block being used to help encrypt the next block.

The ECB mode encryption did a much better job obscuring my personal photo, probably because there were many more colors and variation in the data which made less of an impression. No useful information could be gained from my encrypted personal photo with either encryption mode.

Task 2

I found this task to be quite challenging. Initially I began by trying to use the keys to decrypt the ciphertext and compared the resulting plaintext to the expected string, but I was unable to get this to work. Therefore, I changed my implementation to use the key to encrypt the plaintext and see if it resulted in a matching ciphertext with the expected. I also struggled with keeping track of what was a hex value, what was a string value, and what was a byte value, but the program worked once I was able to sort it all out and made sure I was using UTF-8 encoding to get binary values from strings.

Task 3

2. Given a collision exists somewhere in the data set, it takes 2^m tries to be **guaranteed** a weak-collision match where m is the number of bits of encryption, since you are looking for a match to a **specific** value. To do this, you must compute all possible outputs and compare it to the expected value. We were using 24-bit encryption, so that would be equal to an average of 2^24 = 16,777,216 tries to find a match. It took an average of 21,786,311 cycles for a weak collision to be found by my program over 15 trials.

3. Generally, it takes 2^(m/2) tries to find a strong collision where m is the number of bits of encryption, since you are looking for **any two** matching hash values, not a specific

match. We were using 24-bit encryption, so this would be equal to an average of 4,096 tries to find a matching pair of hashes. My program took an average of 4,773 tries to find a strong collision over 15 trials.

5. Finding a strong collision is far easier than finding a weak collision, per the data collected from these exercises.

6. This is because it is much easier to find any match than it is to find a specific match. Per the pigeonhole principle, once the number of people in a room exceeds 365, it is guaranteed that at least two people in the room will have the same birthday (the number of inputs exceeds the number of possible outputs). However, per the birthday paradox, it is not guaranteed that somebody in the room will have the same birthday as you, which can never be completely guaranteed no matter the number of people in the room. Put simply, it is much easier to find **any** match than it is to find a match to **one specific value.**