**Steganography and Secret Keys – Dorough**

**Task 1:**

The first task is decrypting a simple substitution cypher. All you need to do is find the 3 most frequent characters and any string which seems most likely to be the word “the” Once you have this you can manually just swap out letters looking for common words and just keep increasing word size until all letters are substituted.

**Graphical user interface, text, application

Description automatically generated**

**Task 2:**

This task is simply to run the openssl enc command to encrypt messages in various protocols. Not much to see other than a jumble of encrypted text.

**Task 3:**

Here you’re supposed to encode an image using both ECB and CBC encryption methods. However, the lab instructions provide no reference to what either of these encryptions are. So, I have researched what the outcome should be. Below is an example of the difference in predictive ECB protocols and pseudo random CBC protocols:

Graphical user interface

Description automatically generated with low confidence

**Task 4:**

This task is to show what padding value is added to a message when being encrypted using CBC protocols. The image below shows what data is padded to the encryption message: Text

Description automatically generated

**Task 5:**

Here we show the loss of data that can be caused by corrupting a file while it is encrypted. I created 1 file for all, {ecb, cbc, cfb, and ofc} protocols and corrupted the 20th byte on each of the files.

Graphical user interface, text

Description automatically generated

Table

Description automatically generated

From this when you decrypt the files, you will be missing quite a bit of data.

A screenshot of a computer

Description automatically generated with medium confidence

**Task 6.1:**

The primary reason that you should never use the same pairing of IV with a key is that it becomes predictive. Dictionary, wordlist, or other brute force attacks can test for matchings of IVs to find the protocols and values needed to derive they key.

**Task 6.2:**

Here you decode a message by using known information {plaintext 1, ciphertext1, and ciphertext2}. To do this you follow this formula:

Text

Description automatically generated

**Task 6.3:**

Here the task is like the last one. However, the IV changes after each input. You need to use the same formula to discover if the initial plaintext is yes or no. using some code and the previous information provided such as the first actual IV, the ciphertext of the first message, and the next IV. You can return a hex string that either matches your guess or is completely different. Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated