**Task 1:**

Each of the encryption methods all return a 16 byte value:

Matts-MacBook-Pro:Task1 mattpopovich$ python Task1-AES.py

e8da47acc08bc751745ef8fbff44e107

Matts-MacBook-Pro:Task1 mattpopovich$ python Task1-ARC4.py

2ef6f3a3e2cdc744558ef389e16a1eed

Matts-MacBook-Pro:Task1 mattpopovich$ python Task1-Blowfish.py

ab8982ce90b3dd16c843c08a3f091430

Matts-MacBook-Pro:Task1 mattpopovich$ python Task1-Des.py

c41bf1b836627b0ffba38073b1dff6a4

Matts-MacBook-Pro:Task1 mattpopovich$ python Task1-Des3.py

074c3a14f06b70dfd7e13b0a776adda6

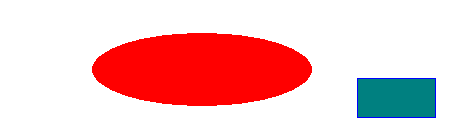
Matts-MacBook-Pro:Task1 mattpopovich$ python Task1-XOR.py

3e1c10170606453915120012100a5225

Although they all respond with a 16 byte value, there are a few differences. The key for Des is 8 bytes long, unlike the others which were 16 bytes. AES ECB needed an instance vector (iv), which had to be the same size as its key.

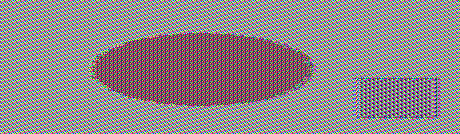
**Task 2:**

I was given the image below in Figure 1 and told to encrypt it with both AES CBC and AES ECB.



**Figure 1:** Original Image

Figure 2 illustrates the encryption with AES ECB. Notice that the image is somewhat identifiable.



**Figure 2:** Image encrypted with AES ECB

Figure 3 illustrates the encryption with AES CBC. Notice that the image is now unidentifiable.



**Figure 3:** Image encrypted with AES CBC

**Task 3:**

This task had me encrypt a 64 byte long text file, corrupt one bit, and compare the decryptions of both AES CBC and AES ECB. The original text used was:

This file is 64 bytes in length and is to be encrypted via AES!!

Below is the AES ECB encryption, corruption, and decryption:

ºc≤\ÄtÚeÙÖ—ø;Cîß>˛k¢ë u•( Gô0Äˆ@3ƒú¡\_uÿ%PSdï­VÓ¸$‚\*ò— ÍG

ºc≤\ÄtÚeÙÖ—ø;Cîß>˛k¢ë u•( Gö0Äˆ@3ƒú¡\_uÿ%PSdï­VÓ¸$‚\*ò— ÍG

This file is 64 mù›v™0˘M±X0riÛêand is to be encrypted via AES!!

Below is the AES CBC encryption, corruption, and decryption:

"."ü@:Ö¬H„ÇQiÄlÎgﬁ◊:MN>ê=Ú‑Æ®´2's◊6ÔôfÅ∆÷˘Ûá~y≠≠Å(D{op[íõ

"."ü@:Ö¬H„ÇQiÄlÎgﬁ◊:MN>ê=Ú‑Ø®´2's◊6ÔôfÅ∆÷˘Ûá~y≠≠Å(D{op[íõ

This file is 64 ƒtz«# o3ﬂàßi!and is to be dncrypted via AES!!

What to take away from this is that one corrupt byte corrupts a whole block of decryption. With AES CBC it even corrupts a byte outside of the corrupt block.

The reason for the corruption is since the corrupt byte is the 30th byte and the blocks are separated into 16 bytes, the whole second block is unreadable/corrupt. However, while this is true for AES ECB, there is more to the story with AES CBC. With CBC the second block’s encrypted cipher text is XOR’d with the third block to finalize the decryption. Because of the 30th byte’s corruption, this byte error spills over to the third block as well, but only for its (30-16=) 14th byte.

**Task 4:**

Task for we are given a dictionary of possible keys, and told to find which one was used for the encryption. The procedure was to loop through all keys, comparing the encryption with the correct encryption. Cleverly, the correct key was “hack”.