COMP 7402 Assignment 5 Report

Peyman / Dimitry

Goal

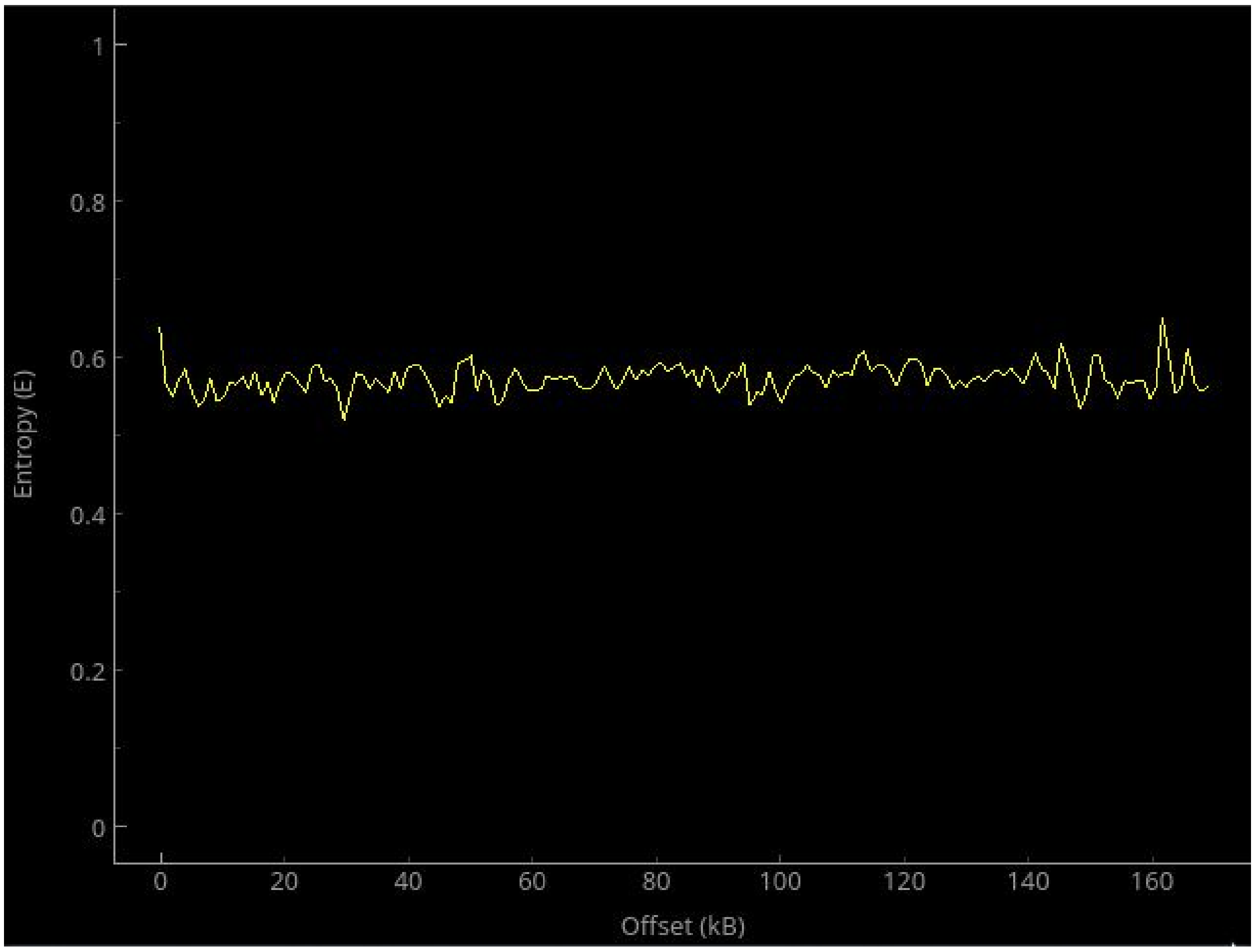
For this assignment our goal was to create a simple Feistel cipher with 8 rounds. The cipher would provide an option to use either ECB or CBC encryption schemes. Another feature that would be added to the previous assignment would be sub-key generation.

Analysis

This implementation causes the ciphertexts to be rather random. During testing we found that to the naked eye the output of the ECB function appeared less random than the output from the CBC function. This had to be confirmed using binwalk's entropy setting.

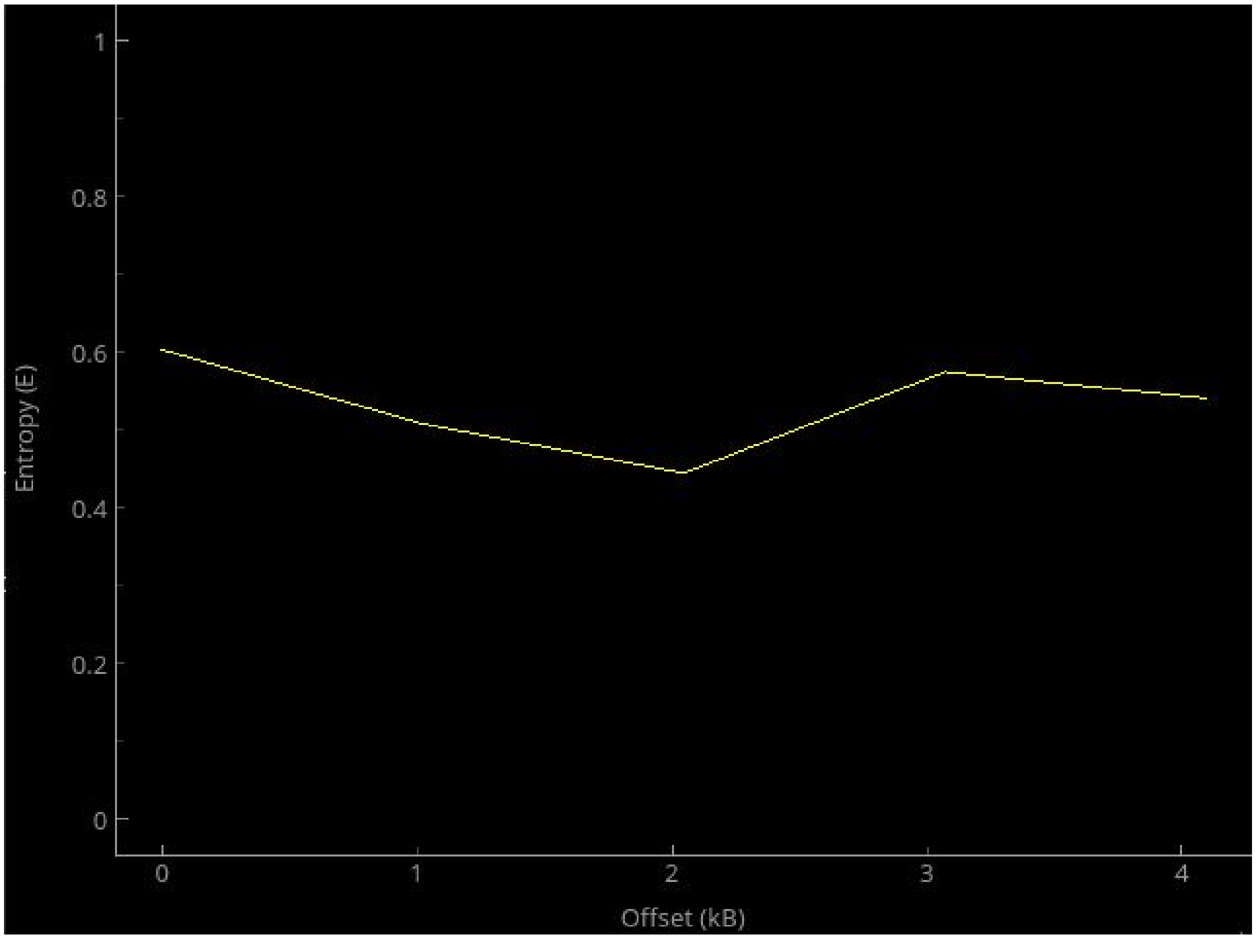
Alice Entropy

First it would help to take a look at the entropy of the english language. Taking a rather large sample such as "Alice in Wonderland" we can see that the entropy is around 0.6.



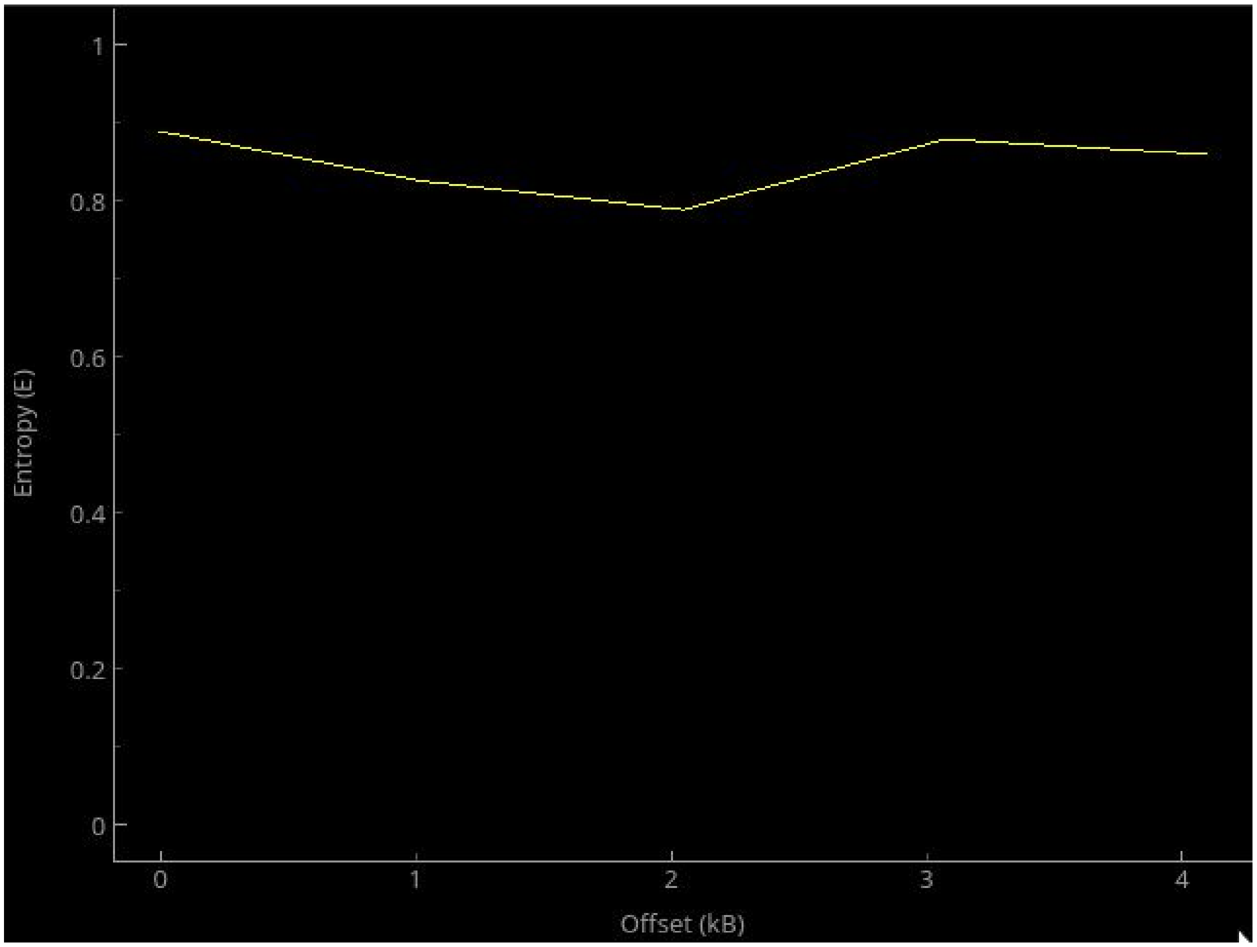
Source Code Entropy

Following this we can attempt to analyze the entropy of out plaintext file src/feistel.c. This file comes out to have a roughly similar entropy to the large text file we tested above.



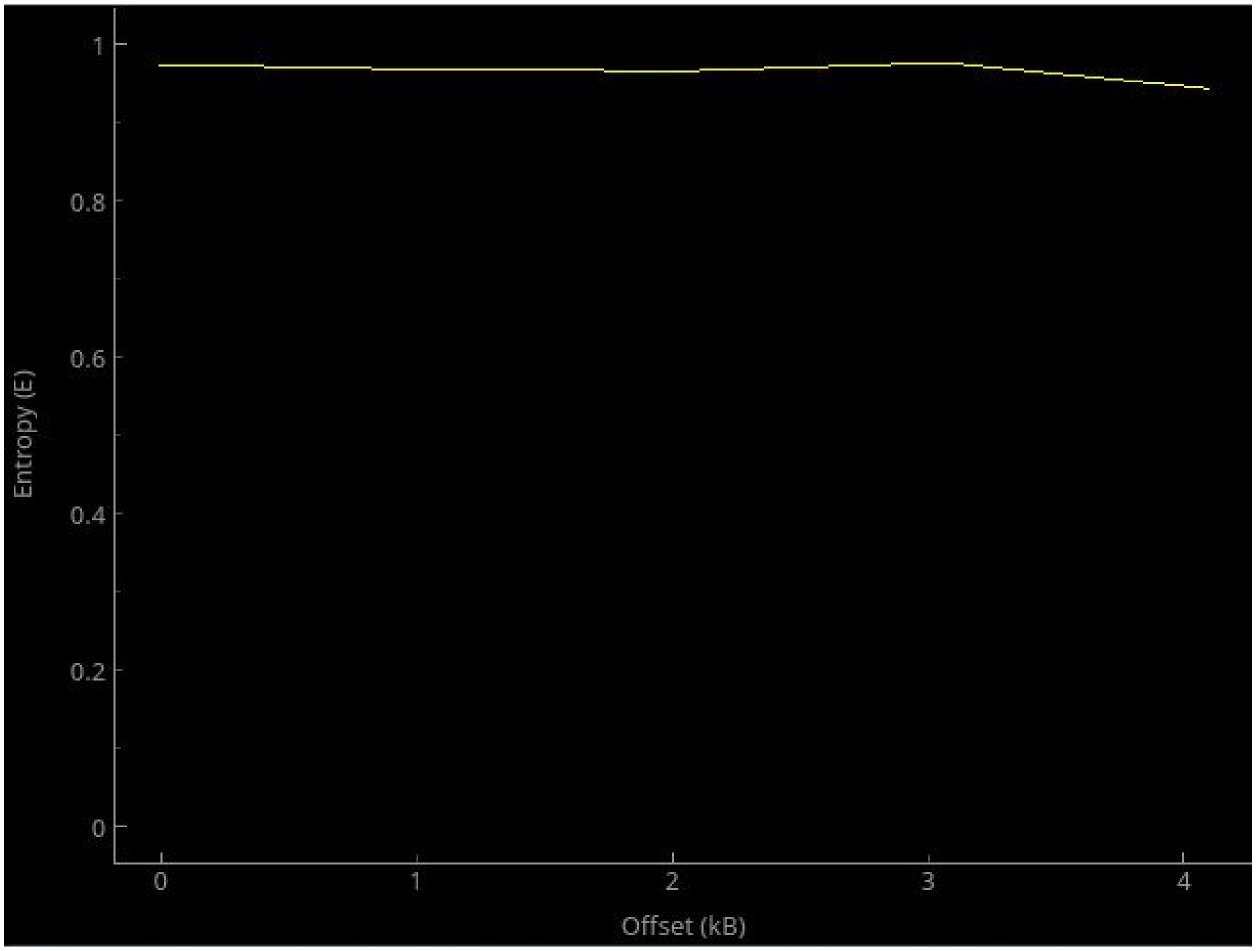
ECB Encrypted Source Code Entropy

Using this new found information we tested the output file from the ECB function and found that the entropy is much higher than in a plaintext file. This shows that the diffusion of data is much greater because the randomness of the bits has increased greatly.



CBC Encrypted Source Code Entropy

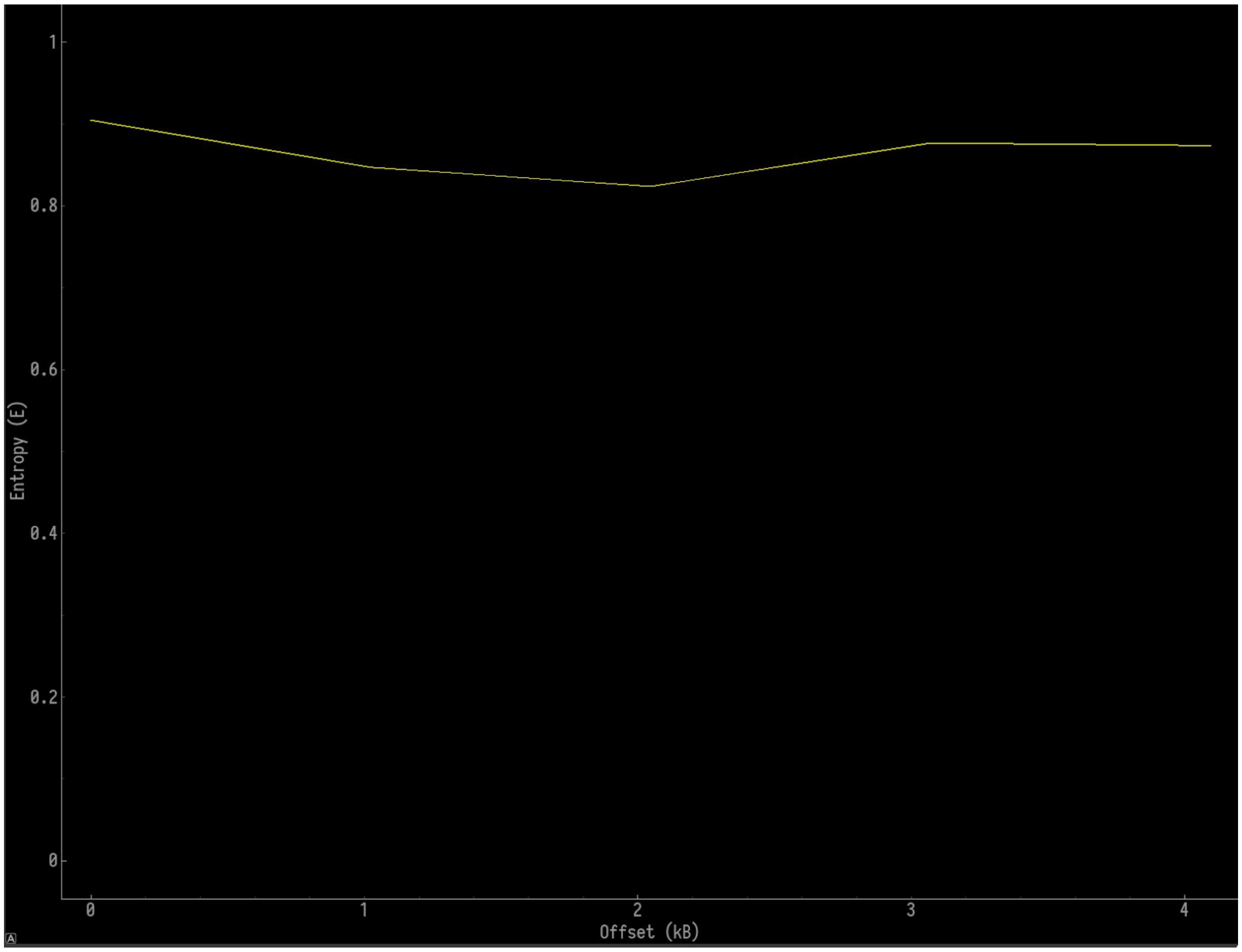
The results of this test were rather unexpected. The entropy of the same file encrypted using CBC was much higher than that of ECB. In face the entropy was almost 1 which was rather impressive. This is probably because it introduces additional entropy from using the previous block cipher text. One could almost not distinguish the data from a block of data generated from /dev/urandom in the next section.



Entropy (E)
Offset (kB)
0
0.2
0.4
0.6
0.8
1
0
1
2
3
4

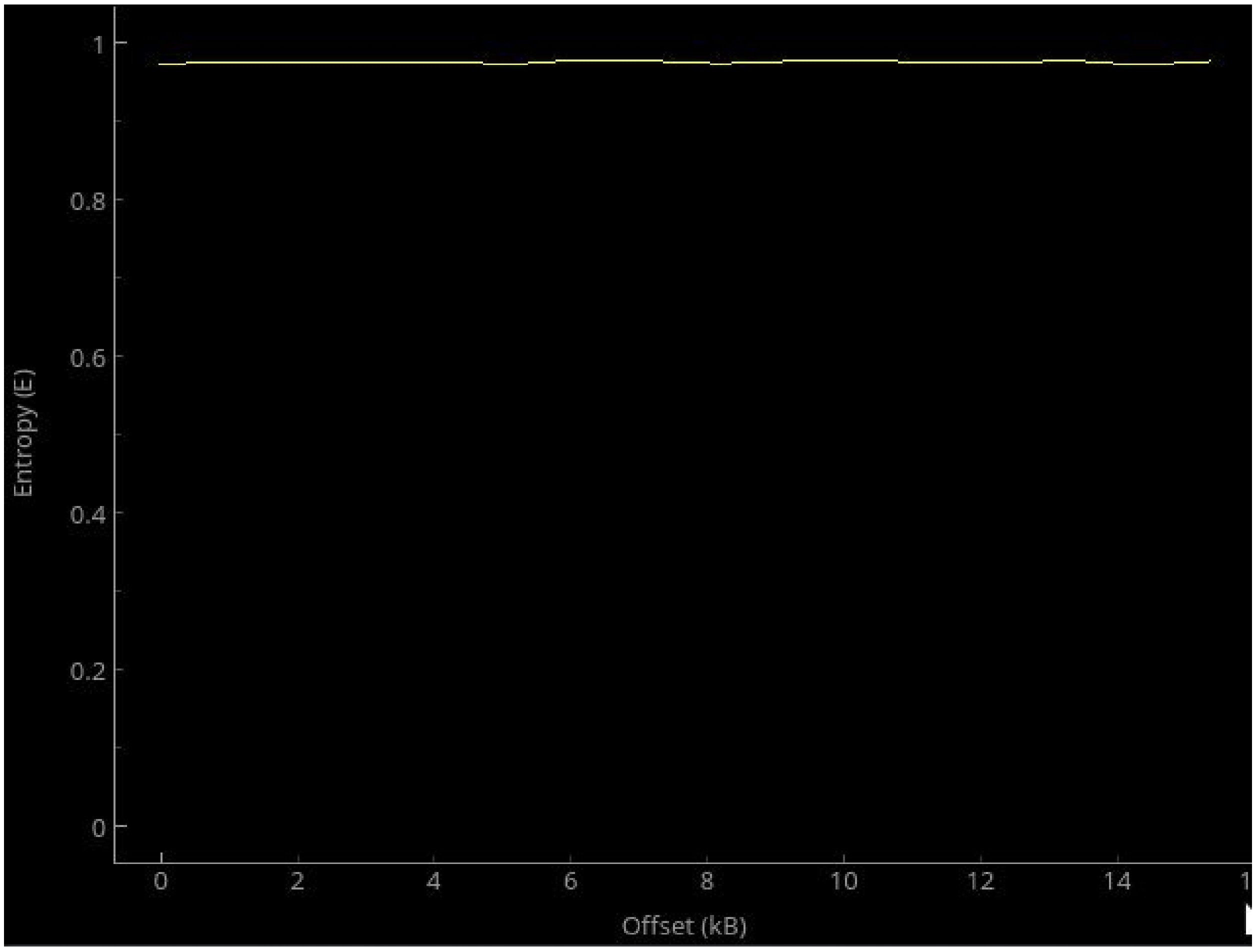
CTR Encrypted Source Code Entropy

CTR encryption didn't have as high of an entropy as CBC but that is probably due to us using the same nonce for every block.



Entropy of Data From /dev/urandom

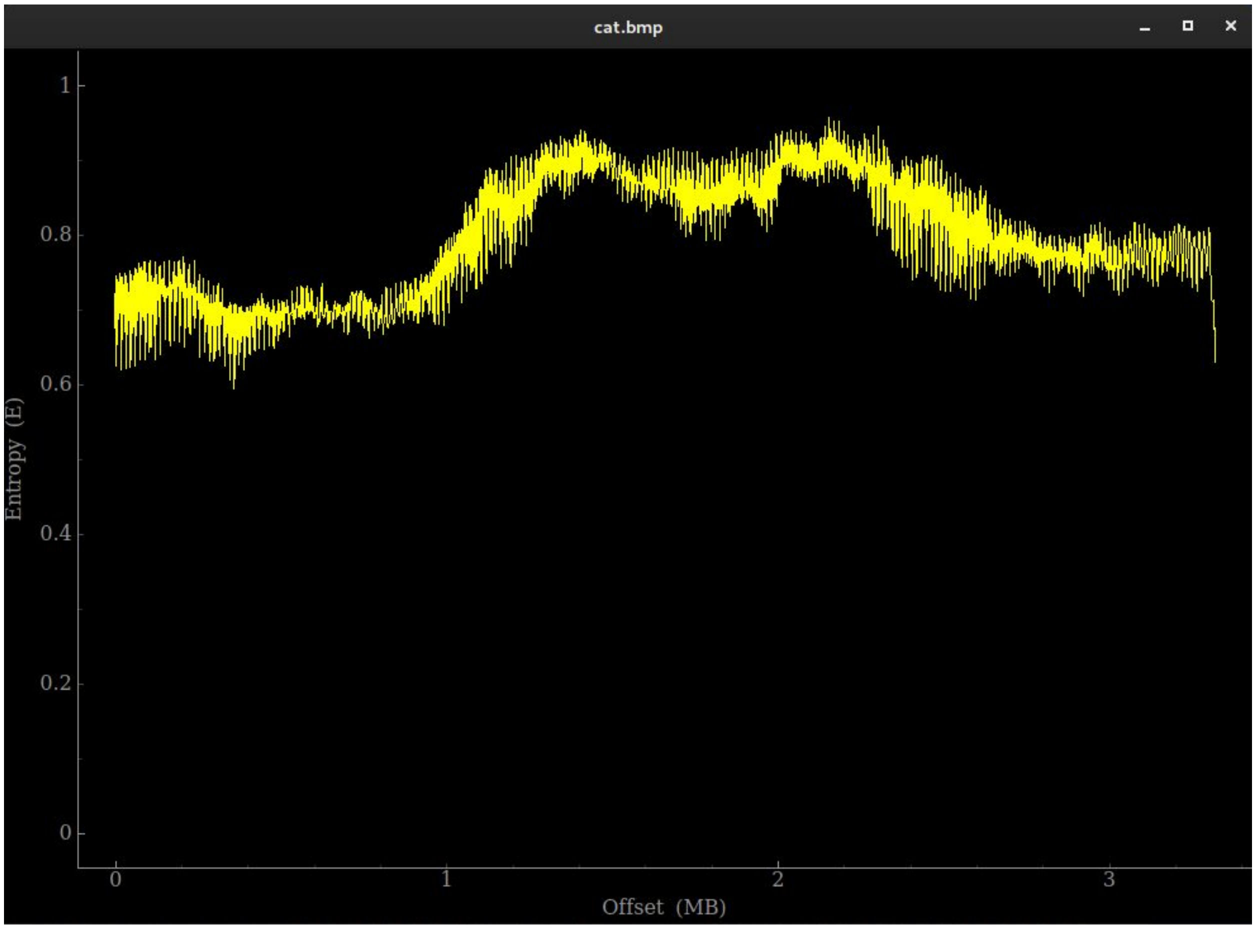
The entropy of urandom was checked to see how high the entropy is. Here we see that while it is much more stable in the range of ~1 it is still very similar to the output of the CBC mode.



Visual Analysis

Now we will analyze an image called cat.bmp by looking at its entropy and the image itself.

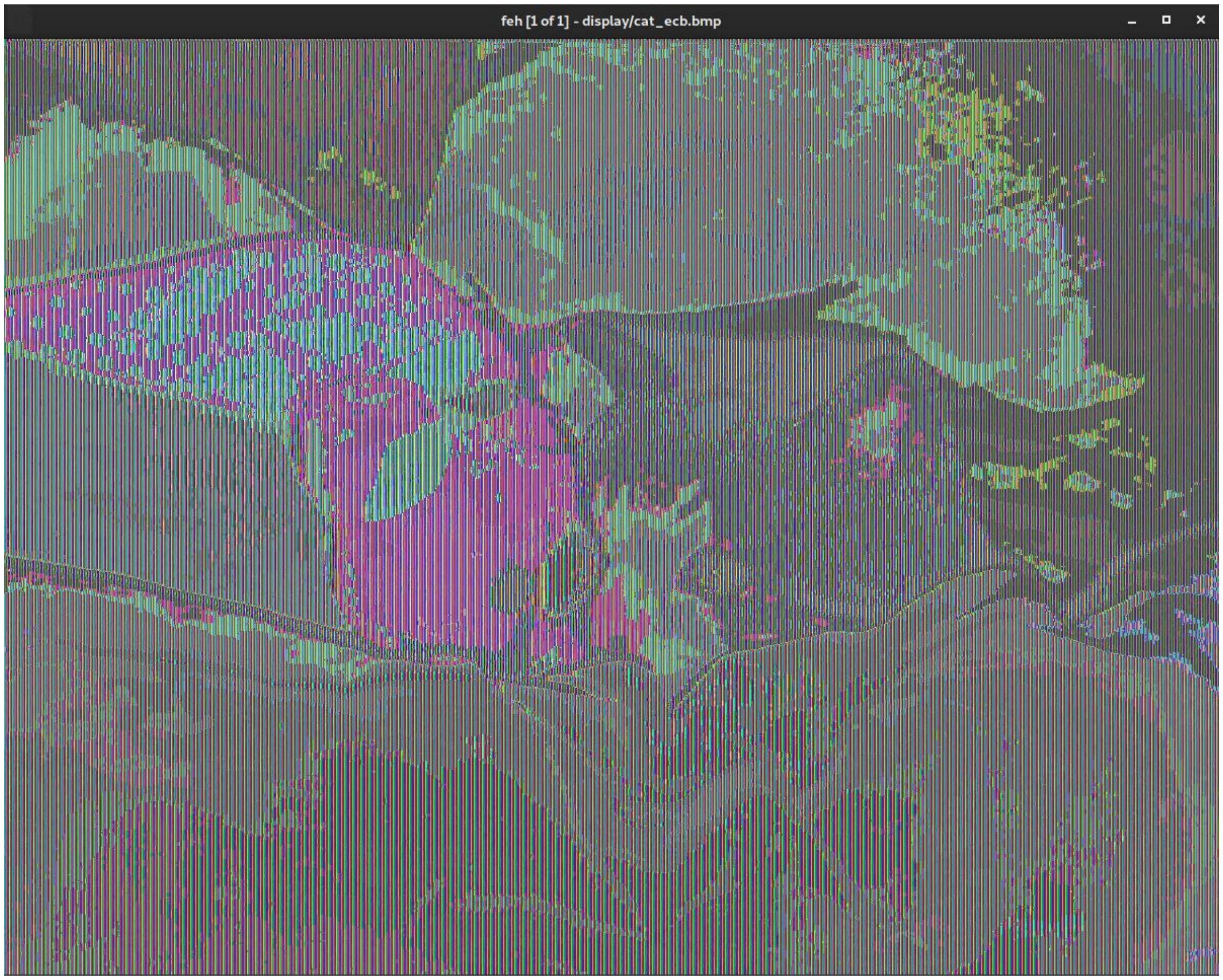
Source Image



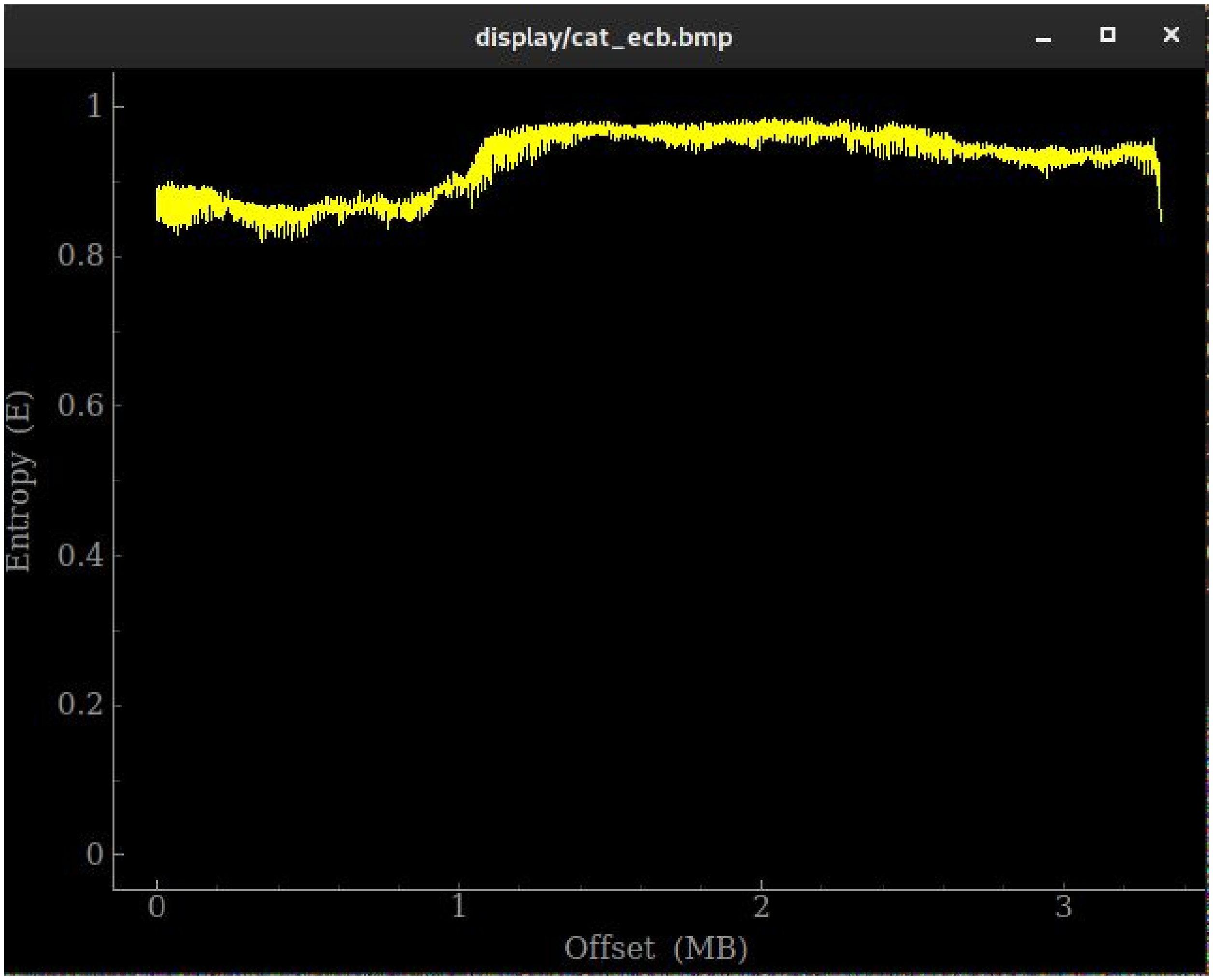
The entropy is higher than that of english text but it is also not completely random. We can also see a great deal of variation on the curve itself.

ECB

Now the ECB version of the cat picture looks like the following.



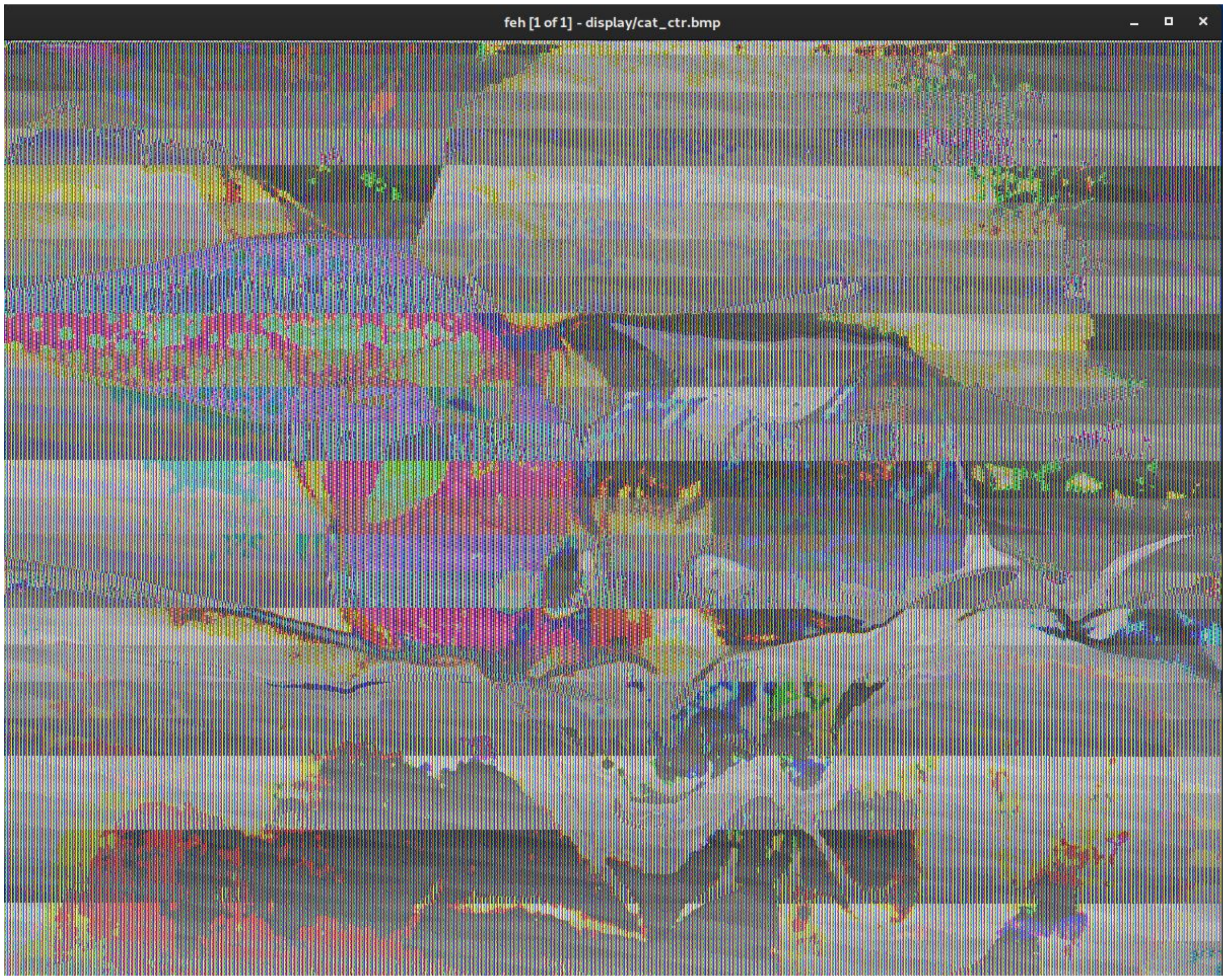
We can see that the 64 bit blocks are mangling bits inside each block but the damage is not sufficient. The discoloration appears to be consistent because the mangling is done in similar patterns. The data representing red green and blue pixels is moved around. The mangling is not propagated through the image and the overall picture can still be seen.

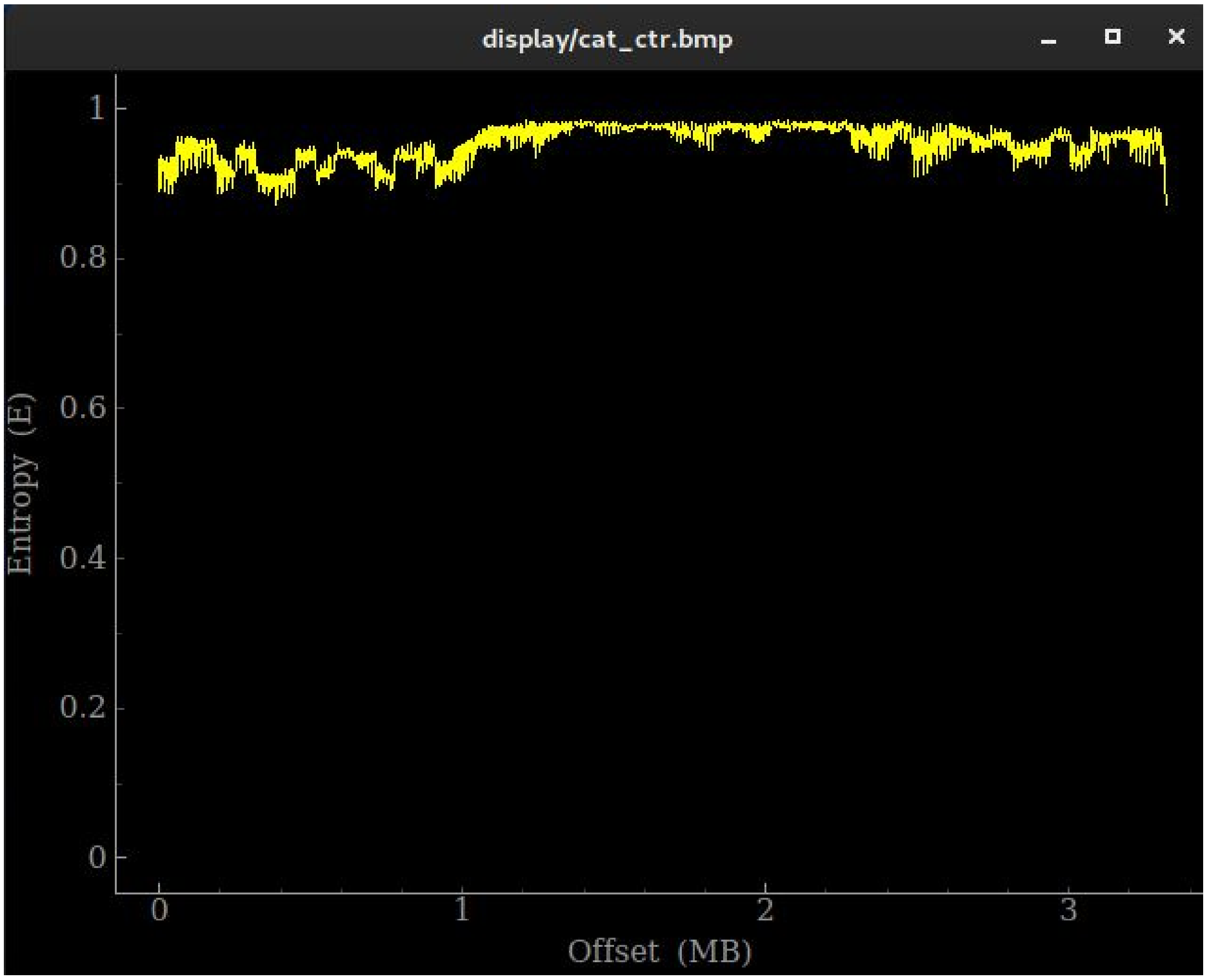


The entropy of the encrypted image is a bit higher but has even sharper variation. The entropy here is on average 0.86.

CTR

The CTR data appears to be modified based in blocks but also in rows of the image. The extra layer of modification is due to the counter variable being incremented after each 64 bit block.

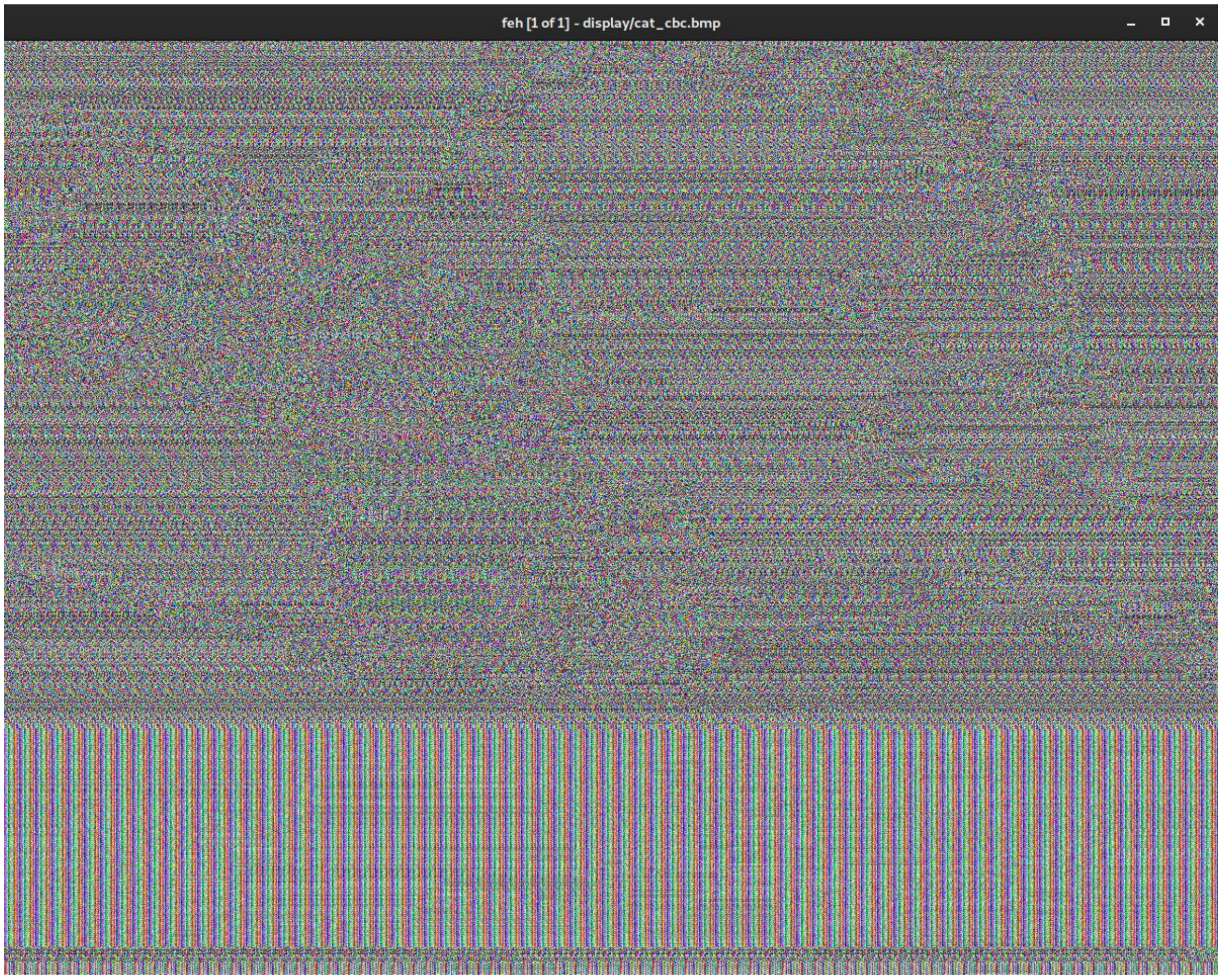

feh [1 of 1] - display/cat_ctr.bmp



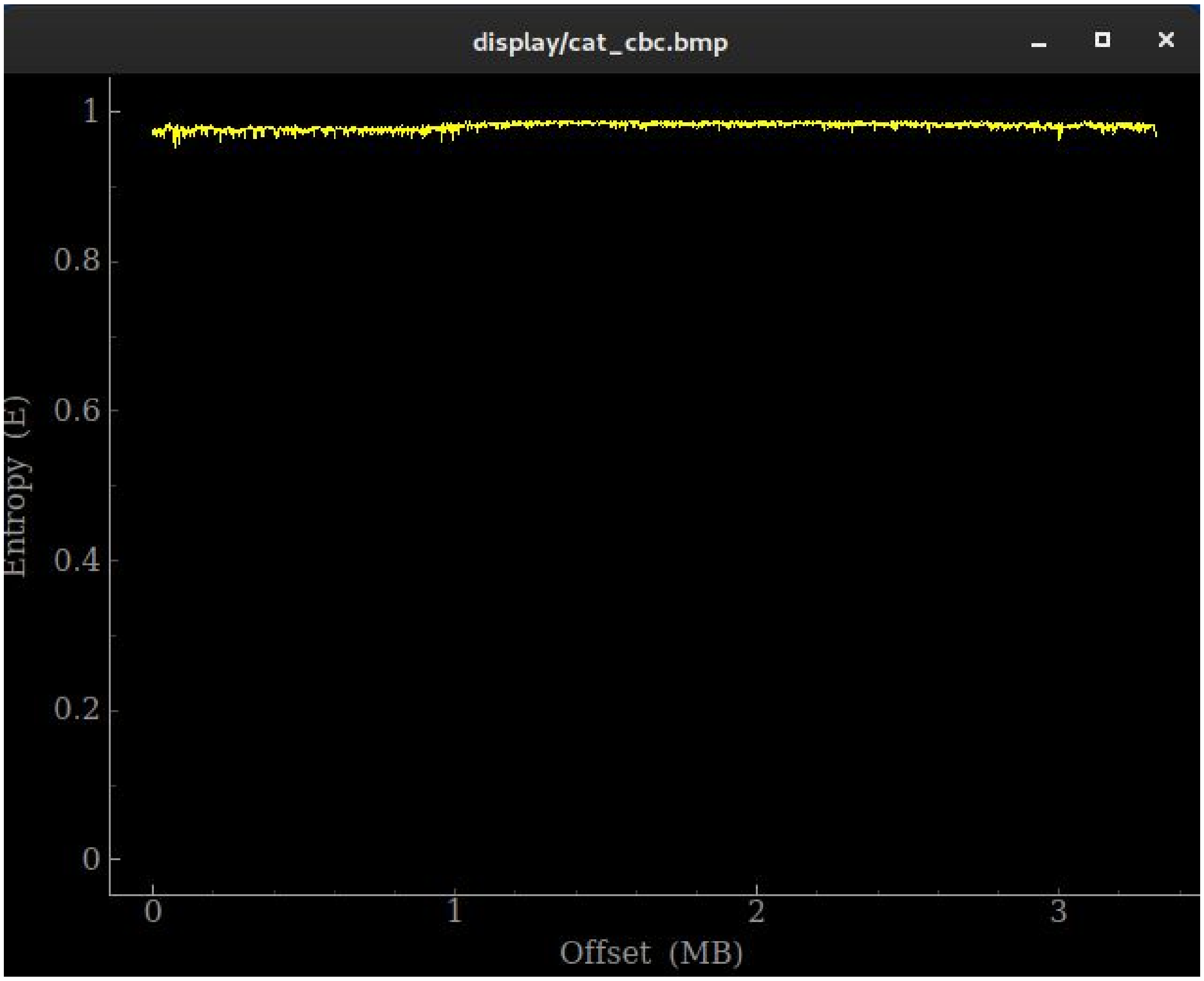
The entropy of the image in CTR mode is higher than that of the ECB mode image. Mainly because it has the additional changes being made via the counter / nonce. The entropy on average is around 0.95.

CBC

The image after encryption looks completely random apart from a few seemingly repeating patterns.



This is by far the best result of all the previous attempts.



The entropy is consistently high at around 0.976. This is due to the previous block's ciphertext affecting the following one. The randomness generated is truly impressive.

Conclusion

Due to the evidence collected it appears that CBC mode is much more secure than ECB mode as well as the CTR mode. This is probably the case because in CBC mode the previous block affects the encryption of the each block following it. Both the ECB and CTR modes only affect the current block and therefore in total the data comes out encrypted in local blocks but the whole picture keeps patterns.