Lab 1

Ally Smith (Section A)

March 5, 2022

1. Encryption using different ciphers and modes

I tried the AES-128-CBC, DES-CBC, and DES3 encryption modes, each with a few key values. For each of the encryptions I made, even a small difference in the key made a significant change in the cipher text, demonstrating the avalanche effect.

2. Encryption Mode — ECB vs. CBC

I encrypted the picture using they key and initial value from task 1. After fixing the header to the bitmap file, I was able to see no patterns in the encrypted image, seen below. The image that was generated looks to be random 'noise,' so no useful information can be derived from the picture.

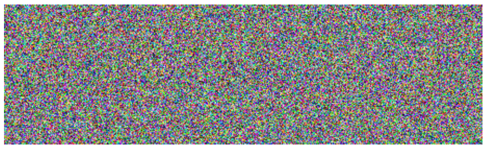

Figure 1: Ciphered image

3. Encryption Mode — Corrupted Cipher Text

I suspect that using ECB on the corrupted cipher text will have the most recoverable text, as there is no feedback, so the bit error will not propagate beyond the block it is in. In CBC, the previous block's cipher text is XORed with the decrypted block, so the bit error will affect the decrypted text in the block *after* that which the error occurred in the cipher text. In CFB, I suspect that the error will greatly affect the deciphered text, as the ciphertext block is used as feedback to the next block. Because feedback in OFB is independent of the plaintext or ciphertext, it is possible that a small (1 bit) change in the ciphertext will not affect the decryption process at all.

After performing the encryptions, changing a single bit in the 30th byte, and decrypting, I was surprised by some of the results. For all of the methods used, over 90% of the plaintext was recoverable. Below is a table showing how many had changed in the decrypted plaintext from the plaintext (out of a total of 446 characters).

ECB	16
CBC	15
CFB	18
OFB	0

4. Padding

5. Programming using the Crypto Library

6. Pseudo Random Number Generation

(a) Measure the Entropy of Kernel

(b) Get Pseudo Random Numbers from `/dev/random`

(c) Get Random Numbers from `/dev/urandom`