Colin Grundey

Principles of Computer Security

Programming 2

Task 1: Encryption using different ciphers and modes

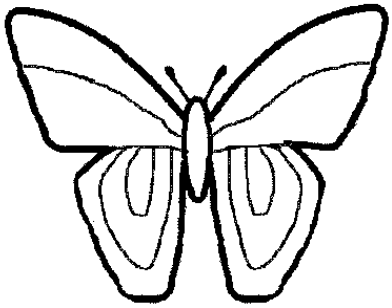
```
seed@seed-desktop:~/Documents/p2$ openssl enc -des-cbc -k "ColinKey" -iv 0 -in e
ncrypt_this.txt -out descbc.bin
seed@seed-desktop:~/Documents/p2$ openssl enc -des-cfb -k "ColinKey" -iv 0 -in e
ncrypt_this.txt -out descfb.bin
seed@seed-desktop:~/Documents/p2$ openssl enc -des-ede -k "ColinKey" -iv 0 -in e
ncrypt_this.txt -out desede.bin
seed@seed-desktop:~/Documents/p2$ openssl enc -aes-128-cbc -k "ColinKey" -iv 0 -
in encrypt_this.txt -out aes128cbc.bin
seed@seed-desktop:~/Documents/p2$ openssl enc -aes-128-ecb -k "ColinKey" -iv 0 -
in encrypt_this.txt -out aes128ecb.bin
seed@seed-desktop:~/Documents/p2$ openssl enc -aes-128-ofb -k "ColinKey" -iv 0 -
in encrypt_this.txt -out aes128ofb.bin
seed@seed-desktop:~/Documents/p2$ openssl enc -rc2 -k "ColinKey" -iv 0 -in encry
pt_this.txt -out rc2.bin
seed@seed-desktop:~/Documents/p2$ openssl enc -rc2-40-cbc -k "ColinKey" -iv 0 -i
n encrypt_this.txt -out rc2-40-cbc.bin
seed@seed-desktop:~/Documents/p2$ openssl enc -rc2-64-cbc -k "ColinKey" -iv 0 -i
n encrypt_this.txt -out rc2-64-cbc.bin
seed@seed-desktop:~/Documents/p2$ ls
aes128cbc.bin  aes128ofb.bin  descfb.bin  encrypt_this.txt  rc2-64-cbc.bin
aes128ecb.bin  descbc.bin     desede.bin  rc2-40-cbc.bin    rc2.bin
seed@seed-desktop:~/Documents/p2$
```

Task 2: Encryption Mode – ECB vs. CBC

Commands used:

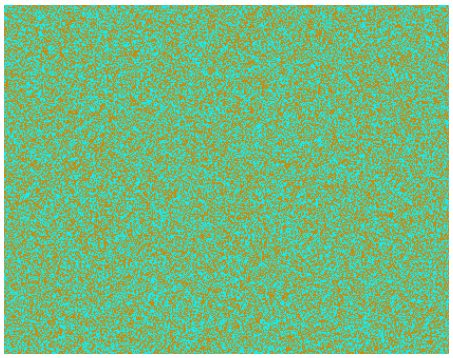
1. openssl enc -aes-128-ecb -e -in pic_original.bmp -out task2ecb.bmp -K 00112233445566778889aabbccddeeff -iv 0102030405060708

2. openssl enc -aes-128-cbc -e -in pic_original.bmp -out task2cbc.bmp -K 00112233445566778889aabbccddeeff -iv 0102030405060708



Original

ECB



CBC

When using AES-128-ECB encryption mode, the bitmap image is somewhat distinguishable when restoring the original header to the encrypted file. However, AES-128-CBC mode does not leave any trace of the original image after restoring the original header. This shows how ECB is not as strong an encryption mode as CBC is due to the chain method used in CBC.

Task 3: Encryption Mode – Corrupted Cipher Text

1. ECB: 14 incorrect characters
   CBC: 16 incorrect characters
   CFB: 17 incorrect characters
   OFB: 2 incorrect characters
2. ECB and OFB have the fewest incorrect character count because they are the least secure modes. ECB doesn't have any feedback for a chain encryption technique. OFB has a simple output feedback for subsequent block ciphers. CBC and CFB are a little more involved, causing the algorithm to fail more when a single byte is modified. Their feedback mechanisms allow the algorithm to generate a more secure encryption, thus the decryption isn't as successful with ciphertext that has errors.
3. The less errors there are means that two files that differ in one byte can be decrypted to a very similar output. This shows the weakness in the modes because the output should differ more given even a small change or error in the ciphertext.

Task 4: Guessing the Key

I wrote a bash script to guess the encryption key which turned out to be "median". The execution time is about 3.5 minutes which isn't very long considering the security that AES should be providing. However, the information we were given to solve this was very helpful and not usually known when doing this. So my conclusion is that brute forcing this decryption was easy with the information given, but will not always be this simple and successful.