HW02 ECE 40400

Matteo Miglio

February 4th, 2021

Problem 1.1 The tools needed to start DES encryption were given to us in lecture 2, accompanied with the understanding of how repetitive Feistel structure encryption works. We start off by taking in an input message of plaintext (a news article of Qualcomm chips and the security problems associated with the chips). Next we take in a key.txt file which contains the key we will be using for encryption within this function. We start off by opening the key file and generating our round keys with the 8 bytes provided (zoomzoom). This creates a 56-bit encryption-key with the last bit of each byte used for parity (first just by permuting the key) and then shifting and then permuting again. Finally we change each round key into 48 bits to be used within the feistel function. Within the actual feistel function, we take 64 bits at a time and break it up into equal right (RE) and left halves (LE). Our right half will go through an expansion permutation, meaning that it will divide the 32 bits into 8 groups of four bits and then take 2 adjacent bits to create 8 groups of 6 for a total of 48 bits. After this, we XOR these 48 bits with the 48 bits from the round key. Next, from each group of 6 bits, we take the outer side bits and use those for row indexing ($2^2 = 4$ rows [0:3]) and then take the inner four bits to index into a column in the S-table ($2^4 = 16$ columns [0:15]. This turns our 48 bits into 32 bits again. Finally, we go through one last permutation using the p-box ordering. After all this, we XOR these modified right (RE) 32 bits with the unchanged left (LE) 32 bits. This XOR result becomes the new (LE) for the next round and the previous (LE) becomes the new (RE). This will happen 16 times for each 64 bit block of data in the plaintext file, each time the only thing changing is the round key.

Problem 1.2 Decryption works the same way as encryption due to the properties of the XOR operator. The only thing that changes is that the round key ordering must be reversed and we must read in the encrypted file as hex and not ascii. The decryption needs the same key as the encryption function. The decrypted file should result in the plaintext file.

Problem 2 This encryption involved image DES encryption. This followed the same exact process as problem 1, exceptions being that we had to account for the three lines of header in the beginning of the ppm file that holds the encoding for the rest of the file.

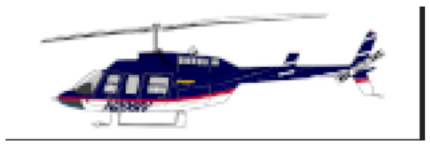

Figure 1: An image of the normal ppm

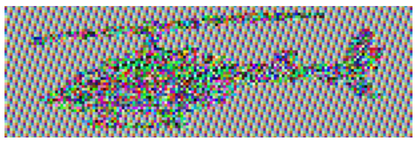

Figure 2: An image of the encrypted ppm file