HW2
Vivek Khanolkar
vkhanolk
2/4/2021

Problem1:

For this problem I implemented DES based on the Feistel function. To achieve this I followed along with the steps in the Lecture 3 notes, as well as utilized code from the Lecture 3 section as well. The steps involved :

- Getting the encryption key
- Generating the round keys
- Reading in the file as a BitVector
- Running the feistel function 16 times
    - Permute the right side
    - XOR it with the round key
    - Perform S-box substitution
    - Perform P-box permutation (matches which input bit is corresponding output bit)
    - Add two halves together again
- Perform one extra swap to realign again
- Convert bitvector to hexstring and write to file
-

The same code was used to decrypt except I just had to call the round_keys in reverse as well as read in the file differently. Since it is a hexstring I could not use the bitvector read file modules, and as a result used a loop where I would increment reading my bitvector 64 bits at a time much like an array.

For the key ‘zoomzoom’ the encrypted message was:

```
d04a94419ec6556c20029c83a277790c5c6380595291ecc23a40b90d60ae5b114dcefad2a
37652e80dbed6bea5ab59d92b8f043c65e1ced023bfe2aa6c4e162de19db8a75ff0f779baa3
629395da7d740e784fd3150db010f0054cd9f70a13ad6a553f954a79ca990dadc1a697ed88
21548099cadedb2d6572810b06df68150cd16af08948628fab087c8577826ee1e0ca728ea
3def08044613e608e9ee27cf91a7052f2d11e6a42b371c5216e296a5486887d3317945023
00e42cfe9b228da863420c7a9d2eb3797bf08185451fc5948a61890e2fec008abe98af6a31
3ba886300a3041f4ca3f273f177fdd95fb97cfd7724c196421848826c105892bfbbe47e6455
1e146fc6d7130d00a2dd01fa6b14a6fb6fb054f843710ddd9a311d54882db94802ceac4fd4
54332747d76b4e6be9e614545db3e6a8517c413628268c07aa64f7175ce8cd40a00a86fb2
```

```
79fed136146b2f863a0f54bb9407a21418641ba55b6641e1acb69bdf816a2cf41479f80ddd
e5a4a43da9f53758f152f58bb5919b65d4a80250c259b38f498f354223cbbcbe14e5408aad
c581eb0d5b19ef8219fbe42edc4e9f0467826a5c1a8141af67f0a897b4f212e5ab49417b576
aba488381be68fc72080ee3ed00b56152e2d7da477b92c98379b694d4f466eb0d93d083fd
62d36ef1ca7f3b4399af80559ffbe0bd48b6eb441a569d479f94a54cb9ca816990971e2298
31db528e70972cae2f82df38026db9db5b118ff17df3a7621911b51626ab948dd95a777b4
219b0ad0ab6180def71f24b42b23444d03b974681d583e07040d443d9365241e1fa77e1b
4684da92913a6ea9a2af407d586ddf8b242706e8775ced9fa520291bbafe441dfab3c4b5d9
3cfe1654202d0b7ff5c381a6a2c489e2c756eb40b6b98482a49878d04f4422fcf43605826d
d6dc32cd8679e51bc800e3ae48673c19c5890c7eec8fc58775299ea756be20afff89395ac6b
021f5bb37c36e30f5948979c96b76537d8785721f1b9789e325d2e779c4e0859c093ba756
c8998219cfc497f0b7f66e259eebea3fba7a9ceed545ed833506d558c2dd9a8812ed9bdc69e
9b0bdfbd514399d2a43be6bf50a2ddab68b3c3b449a430efdd46755871a8697737a7fd251
de37390186a0c701ef7839a2b2ee99a8d6aaf540aefd111c8507120fbc296ade7e0a30846b
4ad461f2af4db654e8e0008fa5a2fa42381d8350bd431d714c42f478ca43e3f31d4e2b77c4f
ea7b5fc92b55c18fd29ac06b78797333758a54e7aab3439dc079d168b7c416e23cd49084a
57ff1c0974dd36102983521b30ecd7fd1931201daab5059b8139f5a3017cd7fd1931201daa
b744fae27721dad95cd7fd1931201daababcce6cc5c4ddacecd7fd1931201daab462cde434e
c9b646cd7fd1931201daab0013da3321192b13b1bcd34158bc5878e3dbd126d0b7edf4cb3
45a27fa36e8df45ed30ec1b4cf954fd1fb2eb165e3fde33aab34a81ef30b95855c1917ea182
6f0093dfae7a9e6124ac8036677dc75ddb8cc79548b5f6b673e2aaa2e74de559d17d4c3597
eb793828ce2eba373130b87f5201f6e90c06758f
```

The decrypted message was:

Smartphone devices from the likes of Google, LG, OnePlus, Samsung and Xiaomi are in danger of compromise by cyber criminals after 400 vulnerable code sections were uncovered on Qualcomm's Snapdragon digital signal processor (DSP) chip, which runs on over 40% of the global Android estate. The vulnerabilities were uncovered by Check Point, which said that to exploit the vulnerabilities, a malicious actor would merely need to convince their target to install a simple, benign application with no permissions at all.The vulnerabilities leave affected smartphones at risk of being taken over and used to spy on and track their users, having malware and other malicious code installed and hidden, and even being bricked outright, said Yaniv Balmas, Check Point's head of cyber research. Although they have been responsibly disclosed to Qualcomm, which has acknowledged them, informed the relevant suppliers and issued a number of alerts - CVE-2020-11201, CVE-2020-11202, CVE-2020-11206, CVE-2020-11207, CVE-2020-11208 and CVE-2020-11209 - Balmas warned that the sheer scale of the problem could take months or even years to fix.

Problem2:

For problem 2 I utilized the exact same DES method as in problem 1. The only thing I had to do was read in the first three lines of the plaintext ppm file (as ppm files have a header that does not need to be encrypted/decrypted) and write it to the output file before performing the encryption.

Here is the encrypted image: