Homework Number: 2
Name: Steven Rutledge
ECN Login: rutleds
Due Date: 1/27/22

*Note: My code runs with “python3” followed by the CL script.

Problem 1:
I have a main function that reads the input and calls either a encrypt or decryption function. These functions use the get_key, get_round_key, and substitution helper functions from the lecture and sample, however they required some changing to work for this program. The encypt pulls out each block, then cycles them through a loop of round keys then prints the final string in hex to the output file. Next the decryption was different because I had to read in hex then index 64 bits out of that in order to decrypt. Also the round keys are revered as the model in the notes says. Lastly I had to strip the padded ‘0’s that we added in the encrypt before writing to file.

Encrypted:
c6d80665196641a1bba15e161d81395b152197d2dcade2d203105c559383da6a81892f0cfba8f3
49aaab0edfbbc420a47af2e1fce559a2163a520e9fc00211e2f73ea73d511cd1e17dc11052e31ed2
996e755b2af37e79abcdb0f0989680d1ccde13f9858462d0575139cdd53c61d0335132ad498d7b
81c1cb8217b731760c78104d829cbf0ece166b0fe2d15cc49dc863716fdafab205e3642c3cbfbdc1
339c9ef905b38e2e36b2c4dfda1bc00d31c0ebea00311a47c6e54d7642f8d3de4396fe1ba0ee330
9585713612c1351a8546bb0b18dd66d2dee64554d22046309e59699975d414e1d4dd2ffc1d052
af0f5ae10392195cf8351a6d23010e07750ef4e1c6ed16d52fafa8d632b1cdd20a4051ec5961a50f
8ecbdaeefce77b9c83effe16af00c301fab16eef790a756a3cb6b97fdcb49467fa265a119ba817922
31ae15ab97140aad0e1e2e2843feaea9723793f4ae46ee68884ce855e0de291f63f723e32058fb5
37b99435db0035e0ea8b0a44516d3c07860c11f31ce612331b1d32dd0df5aef5c75f59a8a16b3e7
43081f4da8a70a3b7540061ca8a741b39d1d54f26aa9b205d59f1e6f72a2a849ea3c3af9f0434f8e
cac2aa556b0a77208a50eed8fd7df099e5956c3bc88137c28a3ebcf4baa189b616987314484643c
d7f23bb9d314d4495f096d918131c6125ea36ecea4d24fd56b1caf927fb9e2ac224bca4d1907e54
ef31c4f5e39bb9c6bc3b60bd6dda0d2c1e1983d81952585909b01ed9020148ef776977cde08984
95cdefd509fc654dd5b87c994aa8e122b12caaa80fccb6c9f15656275749ba1bfe006aeab6993c92
cd409c7fb28f6feb345590730a0ac970872ad95895f69673b328760b4c1e1d58e28539e117d8f28
8e18c5da7eb84e7db47f68ddc87bcc02aeb27e15ae90ed20ecfe9c827073e6d7c56e15cef4d3565
f15a7cba6a36f8e7d9e1ee51ddebd3d0a8bc5a7b3fdc8e4aa4113c6ebe6c7bc0c2f1b2afe5159efbf
d9b6e03124422dcb9c476a1f1ab40c796e12af3066982330065b18b7e1149b7144db2f03b4c876f
5139370752d121eb78e6f20c409b0e92e555cf6d87907dd50222e5531a37c58dcc61c630cbfcc0d
53f61bddce64f2178fabdf7b02f279d3f1d8d2103c70909ec5fe143cc9aaeef07fe323cb803646cc5
c687bbe5d1eb071a72ce6d92601e80c2e7290bb1313ceaa93c2af42fd8b4f430aa371eb123e490b
cff9e18d4407db1607f3b655e02b0883556d40f4edf3d63439f91de6112e388f534be43e6c54017
8f58d54ebd0be617235e6c8d2d9e0850a46f51e168993973532f7e97d2990063788531d0c0831b
cb232f7e97d29900637cefc0c7b39ebd4dc32f7e97d29900637fc33e6e02f4bafd532f7e97d29900

6374bca1971770d1a6632f7e97d299006371bb599c88aec1d163844e9837a4ba27c4e805bd8017
d74050ec365b6e9192729ecd9443195fadea4dfeb06aefe0c2cb3b8fc0ce0e806d6df42d85a4cef1
c2852fcd77dc415d494382cd8dfbd854b2578c8f0970e8bc5e57264a3ef16b730afd81235ae6605
3429bb9323e912d22b7d06e19c4b3c9259319d

Decrypted:
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.

Problem 2:
My code uses the same DES algorithm as problem 1 however it is set to read and write the file differently because it is an image. I set up a Boolean called check to just identify if the header has been pulled yet. I found the header to be 128 bits so I read those and stored and wrote them separately. Then in a bv if more to read look I used the DES algorithm described above on the image bits. I then converted it to hex then bits to write to file.