Assignment 4

Due: 1st March, 2023

Problem 1

Write a Python script that implements the full DES algorithm. Your script must produce the correct encryption/de- cryption results when provided with an encryption key.

Program Requirements

Your script for Problem 1 should have the following command-line syntax:

```
DES_text.py message.txt key.txt encrypted.txt
DES_text.py encrypted.txt key.txt decrypted.txt
```

key.txt is the file containing the 64-bit DES encryption key in ASCII character format.

The encrypted output should be saved in **hexstring** (i.e ASCII character) format to a file with the name of the final argument (in this example, a file called encrypted.txt).

You are encouraged to take a look at the files in the gzipped archive (taken from lecture 3 of Avi Kak's course website). These files include all of the permutation "tables" that you will need for the homework (except for the P-box permutation). The gzipped archive includes a script for generating the round key, a script for showing the permutation of the encryption key, a script demonstrating the substitution step, and a .txt file with the eight S-box tables.

The plaintext bit size is not necessarily divisible by the DES block size. For the sake of this assignment, your program can pad the last block with zeros if this is the case.

Problem 2

As you already know block ciphers such as DES should not be used in the manner described in Problem 1, i.e., by directly encrypting the data in independent blocks (known as electronic code book mode). Because each block of data is encrypted independently, overall patterns in the data may still be obvious if used to encrypt, say, an image file. This is not readily apparent when encrypting text like in Problem 1 but becomes evident when encrypting an image for example.

Your task in Problem 2 is to demonstrate this with a script that performs the following steps:

1. Takes an image in PPM format (described below),
2. Uses DES to encrypt the image data (only the image data, **NOT** the PPM header as explained below) with a key contained in a text file and
3. Finally, combines the encrypted image with a PPM header (you can use the original image's header) so you can write it as a readable PPM file.

The result should be the encrypted image viewable as a PPM file.

PPM Image Format: A ".ppm" image file consists of a header and the actual image data. The header occupies the first 3 lines of the file, and the rest is the image data (that is the image pixel values) that you want to encrypt. For those interested in a more general approach,

descriptions of the PPM header can be found at:

- http://netpbm.sourceforge.net/doc/ppm.html
- http://paulbourke.net/dataformats/ppm/

Program Requirements

The call syntax for your program is similar to that in Problem 1:

```
DES_image.py image.ppm key.txt image_enc.ppm
```

In this example, image.ppm is the input image you will encrypt, key.txt is the key you will use for encryption, and image enc.ppm is the name of the .ppm file you will write the encrypted image to. You may use parts of your script from Problem 1 for this, though you probably will not use all of it as text may be read differently than image data.

Key to be Used for the Submitted Output:

zoomzoom