ECE 404 Homework #2

Due: Thursday 01/27/2022 at 5:59PM

The goal of this homework is to help you understand the implementation of the DES (Data Encryption Standard).

Problem 1

Write a Perl/Python script that implements the full DES algorithm. Refer to Lecture 3 as it outlines the key steps to implementing DES. Your script must produce the correct encryption/decryption results when provided with an encryption key.

Program Requirements

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

```
DES_text.py -e message.txt key.txt encrypted.txt
DES_text.py -d encrypted.txt key.txt decrypted.txt
```

An explanation of this syntax is as follows:

- **For Encryption** (indicated with the `-e` argument):
  - your script should read the input text from a file called `message.txt` (or whatever the name of the file argument after `-e` is).
  - The next argument (in this case `key.txt`) is the file containing the 64-bit DES encryption key in **text string** (i.e 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`).
- **For Decryption** (indicated with the `-d` argument):
  - the input hexstring file is specified after the `-d` argument, in this case `encrypted.txt`.
  - The next argument specifies the file containing the DES decryption key (keep in mind the key used for both encryption and decryption must be the same).
  - The decrypted output should be saved to a file with the name specified by the last argument, in this case `decrypted.txt`.

You are encouraged to take a look at the files in the the gzipped archive for Lecture 3 at your instructor's Lecture Notes 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.

Remember to parse the command-line arguments for your program using the calling conventions described above. Please do not hard-code the file names into your program.

Problem 2

As you will soon learn in Lecture 9, 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 job 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.

Remember to parse the command-line arguments for your program using the calling conventions described above. Please do not hard-code the file names into your program.

Text and Image to be Used for the Submitted Output

The text and image to be used have been uploaded on Brighspace along with the homework. The text is from a computerweekly.com article discussing the a vulnerability in smartphones with a certain Qualcomm chip (it starts with "Smartphone devices from..."). The image is a PPM of a helicopter.

Key to be Used for the Submitted Output:

zoomzoom

Useful Notes

- Keep in mind you are not required to implement image decryption for Problem 2.
- For debugging purposes, we have also have posted to the homework section a file named `first_round.txt` containing the left and right halves of the first plaintext block after the first Feistel round for the text file mentioned above.
- If you are having problems implementing DES, try to debug just one Feistel round first. That is, create a single-round version of DES and see if your decryption can recover the plaintext. For the purpose of debugging, you can start with a key that is made of all zeros and assume, during debugging, that all round keys are the same.
- Remember that text editors can add additional bytes to file contents. When reading in the key you should ignore additional bytes such as the newline character since the key should be only 64 bytes.
- If you would like to use your own key for testing purposes, remember that the encryption key should consist of at least 8 printable ASCII characters. The same key should be used for both encryption and decryption.
- Keep in mind that the `pad_from_left` and `pad_from_right` methods in the BitVector module modify the BitVector object in-place and do not return anything.

Submission Instructions:

- Make sure to follow the program requirements and submission instructions. **Failure to follow these instructions may result in loss of points!**
- If using Python, please denote the Python version in your code with a shebang line (e.g. #!/usr/bin/env python2).
- As in homework 1, you will be electronically submitting your code and your PDF using the turnin command (see the next section for this command).

Electronic Turn-In

- For this homework you will be submitting 3 files electronically. **If you use any of the code from Lecture 3, include it in the files below and properly reference it in the comments.** Your submission must include:
  - A PDF containing:
    - For Problem 1: a brief explanation of your code, and the encrypted and decrypted output for the text mentioned above using the key provided.
    - For Problem 2: a brief explanation of your code, and a picture of the encrypted PPM image.
  - The file `DES_text.py/pl` containing your code for Problem 1
  - The file `DES_image.py/pl` containing your code for Problem 2

`turnin -c ece404 -p hw02 hw02.pdf DES_text.pl DES_image.pl` (if using Perl)
`turnin -c ece404 -p hw02 hw02.pdf DES_text.py DES_image.py` (if using Python)