ECE404 Introduction to Computer Security: Homework 02

Spring 2024
Due Date: 5:59pm, January 25, 2024

1 Introduction

The goal of this homework is to help you further your understanding of the Data Encryption Standard (DES) covered in Lecture 3 [2].

Before you start the programming tasks, you are encouraged to take a look at the files in the gzipped archive for Lecture 3. These files include scripts for generating the round keys, permuting the encryption key, as well as as performing the substitution step - all of which are crucial in DES.

As always, please read the homework document in its entirety before coming to office hours with your questions. The teaching staff have spent a long time writing the assignment to cover many common questions you might have.

2 Programming Tasks

2.1 Problem 1

Write an Object Oriented Python [1] program that implements the full DES algorithm. Refer to Lecture 3 as it outlines the key steps to implementing DES. Given an encryption key and some plaintext, your program must produce the correct encryption and decryption results.

The two commands below specify the exact command-line syntax for invoking encryption and decryption.

```
1 python3 DES.py -e message.txt key.txt encrypted.txt
2 python3 DES.py -d encrypted.txt key.txt decrypted.txt
```

An explanation of the command-line syntax is as follows:

- Encryption (indicated with the **-e** argument in line 1)
  - perform DES encryption on the plaintext in `message.txt` using the key in `key.txt`, and write the ciphertext to a file called `encrypted.txt`

  - You can assume that `message.txt` and `key.txt` contain **text strings** (i.e. ASCII characters)
  - However, the final ciphertext should be saved as a single-line **hex string**

- Decryption (indicated with the **-d** argument in line 2)
  - perform DES decryption on the ciphertext in `encrypted.txt` using the key in `key.txt`, and write the recovered plaintext to `decrypted.txt`

A skeleton file for `DES.py` has been provided below.

```
1  from BitVector import *
2  import sys
3
4  class DES():
5      # class constructor - when creating a DES object, the
6      # class's constructor is called and the instance variables
7      # are initialized
8
9      # note that the constructor specifies each instance of DES
10     # be created with a key file (str)
11     def __init__(self,key):
12         # within the constructor, initialize instance variables
                                  .
13         # these could be the s-boxes, permutation boxes, and
14         # other variables you think each instance of the DES
15         # class would need.
16
17     # encrypt method declaration for students to implement
18     # Inputs: message_file (str), outfile (str)
19     # Return: void
20     def encrypt(self, message_file, outfile):
21         # encrypts the contents of the message file and writes
22         # the ciphertext to the outfile
23
24     # decrypt method declaration for students to implement
25     # Inputs: encrypted_file (str), outfile (str)
26     # Return: void
27     def decrypt(self, encrypted_file, outfile):
28         # decrypts the contents of the encrypted_file and
29         # writes the recovered plaintext to the outfile
30
31 # drive the encryption/decryption process
32 if __name__ == '__main__':
33     # example of construction of DES object instance
34     cipher = DES(key=sys.argv[3])
```

A couple of things to keep note of while working on Problem 1:

- 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.
- You can expect some null-byte characters in your recovered plaintext should you need to pad the plaintext at encryption time.
- 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.
- For debugging purposes, we have included a text file called `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.

2.2 Problem 2

As you will soon learn in Lecture 9, block ciphers such as DES should not be used in electronic code book (ECB) mode as seen in Problem 1 (i.e. directly encrypting the data in independent blocks) to encrypt viewable media such as images. Overall patterns in the data may still be obvious since each block of data is encrypted independently.

Your job in problem 2 is to demonstrate this characteristic with a script that performs DES encryption on an image. More specifically, given an image in PPM format, perform DES encryption **only on the image data** with the same key from problem 1. **Do not encrypt the PPM header**, as you will combine it with the encrypted image data to produce an encrypted PPM image. The structure of a PPM image file is detailed below.

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 while the image data (i.e. the data you need to encrypt) begins on the subsequent lines following the header [3].

```
1 python3 DES.py -i image.ppm key.txt image_enc.ppm
```

The command-line syntax for invoking image encryption is specified above. It says to perform DES encryption on the image in `image.ppm` with key in

`key.txt`, and store the encrypted image in `image_enc.ppm`

A couple of things to keep note of while working on Problem 2:

- Many parts of problem 1 will get reused in problem 2. It is in your best interest to keep your problem 1 solution nice and tidy so that it can be ported over efficiently.
- The implementation for your problem 2 solution can be made as short as one additional class method in the DES class, and some modification to the if name equals main construct.
- Unlike in problem 1 where you are required to write the encrypted data as a hex string, make sure to write the encrypted image data directly to the file (no hex string conversion), otherwise your final output will not be viewable.

3 Submission Instructions

- You must turn in a single zip file on Brightspace with the following naming convention: `HW02_<last_name>_<first_name>.zip`. Do not turn in files other than those listed below. 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.py` containing your code for Problem 1 and 2

References

[1] Object-Oriented Programming in Python. URL https://realpython.com/python3-object-oriented-programming/.

[2] ECE 404 Lecture Notes. URL https://engineering.purdue.edu/kak/compsec/Lectures.html.

[3] PPM - Netpbm color image format. URL https://netpbm.sourceforge.net/doc/ppm.html.