Problem Set 2

This problem set is due **at 10:00pm** on **March 3, 2019.**. Total Points: **115**

- Remember that the problem set must be submitted on Gradescope. If you haven't done so already, please sign up for CS 302 on Gradescope, with the entry code 9WB58J to submit this assignment.

  We suggest that you perform a trial submission with a zip file prior to the deadline to make sure that everything works for you – you can overwrite that submission with a new one up to the deadline.

- We require that written solutions are submitted as a PDF file, **typeset on LaTeX**, using the **updated template** available on Piazza. You must **show your work** for written solutions. Each problem should start on a new page.

- We will occasionally ask you to "give an algorithm" to solve a problem. Your write-up should take the form of a short essay. Start by defining the problem you are solving and stating what your results are. Then provide: (a) a description of the algorithm in English and, if helpful, pseudo-code; (b) a proof (or proof sketch) for the correctness of the algorithm; and (c) an analysis of the running time.

- We will give full credit **only** for correct solutions that are described clearly and convincingly.

Problem 1-1. AES [10 points]

Consider an AES-128 instance. The key is expanded into a schedule of 44 words.
Note: 1 word = 4 bytes

(a) [5 points] How many words from the key schedule would have been consumed just before the starting of round 5?

(b) [5 points] Which words will be used for the ADD ROUND KEY step at the end of round 8? Take the key schedule $\{w_0, w_1, ..., w_{43}\}$ where w_i denotes the ith word.

Problem 1-2. AES [20 points]

Consider an AES-128 instance to encrypt 3 blocks.
Give the total number of times the following steps will be executed :

(a) [5 points] ADD ROUND KEY

(b) [5 points] SUBSTITUTE BYTES

(c) [5 points] SHIFT ROWS

(d) [5 points] MIX COLUMNS

Problem 1-3. AES [15 points]

Consider an AES-128 instance.

(a) [5 points] What will be the replacement byte for 11110101 in the SUBSTITUTE BYTES step? Use the SBOX provided in the slides.

(b) [5 points] Suppose the input block to the SHIFT ROWS step is :

$$81cfb5166cee12a75c50660b56357832$$

. What will be the output state array?

(c) [5 points] The ciphertext in 3c.txt has been obtained from an AES-128 instance in ECB mode. What can you infer about the plaintext?

Program **Problem 1-4.** AES [15 points]

Implement AES encrytion using the given python template.

Problem 1-5. Government organization puts shillings into sodium. (3) [10 points]

There are a number of interesting properties about DES that allowed it to be maintained as a standard for so long. What is special about the S-boxes of DES, and why are they better than any randomly chosen set of S-boxes?

Problem 1-6. Informed father finds hashing algorithm with Parisian negation. (7) [10 points]

A good intuitive question to always ask about a cryptosystem is the value of each step present within it. What is the point of the final permutation in the Feistel function, since it is neither dependent on the data being permuted, nor the key?

Problem 1-7. Sorrowful cry can follow demon to form tool. (9) [35 points]

In this question, you will write a program to implement the DES cipher. This is a complex procedure, so begin by reading the Wikipedia pages on DES and bookmarking the supplementary material.

A few set of guidelines are given below.

Program

(a) You must include a README file with your program, illustrating its operational details, for instance, what padding you used, how to switch between encryption and decryption, etc. Your program will not be marked in the absence of a README.

(b) Your program will be run in two modes - encryption and decryption. You must provide an easy and understandable way to switch between these modes.
In encryption - you may assume that your program has two files available in the directory it is being run, *plaintext.txt*, and *key.txt*. The files consist of characters, where each character represents eight bits of data. It is guaranteed that key.txt will have exactly **seven** such characters (read clarification below for clarification). Your program must then generate a file called *ciphertext.txt*, which should contain the encrypted text.
In decryption - you may assume that your program has two files similiarly available, *key.txt*, and *ciphertext.txt*. You are guaranteed that *ciphertext.txt* contains a valid encrypted message under your conditions, though it may not necessarily be the case that your program was used to encrypt it (once again, please be very explicit in the README about your choices). Your program must output the correct *plaintext.txt* file associated with the *ciphertext.txt* file.

(c) You must follow the conventions given below.

1. Use PKCS5 padding.
2. Do not parity check the key - you will be given a 56 bit key directly, instead of a 64 bit key.
3. In the Feistel function, do not use the standard DES key schedule. Instead, use the DES key schedule as defined in the last part of this question.
4. Do not implement the Initial Permutation and Final Permutation steps of the Feistel network.
5. You must run DES in Cipher Block Chaining mode with a customizable initialization vector.

(d) The plaintext "romeoandjulietweregreatlovers" encodes under the key "Z8tb;a=" with the initialization vector "abcdefgh" (note that I'm representing binary in ascii characters, but the key itself is really 56 bits), encodes to :

11000111 00110111 11010100 11110000 10000000 01110110 10000010 10010110
10000000 11110111 10101100 11000010 11101110 01100010 01011010 01111001
00010010 10011110 00001010 11100111 11011100 01101011 10001001 10011110
00011011 11100101 10100110 11111100 11110011 00011101 00010000 11100011

The plaintext itself, in binary, is as follows (with padding included, so note that the last three bytes are equal to three).

01110010 01101111 01101101 01100101 01101111 01100001 01101110 01100100
01101010 01110101 01101100 01101001 01100101 01110100 01110111 01100101
01110010 01100101 01100111 01110010 01100101 01100001 01110100 01101100
01101111 01110110 01100101 01110010 01110011 00000011 00000011 00000011

(e) A template has been provided to give you a rough idea of how to proceed. The template is in C++, but a lot of parts of it are easily transferable to other languages (especially the S-Boxes, which have all been filled out in a three dimensional form for you). A few lines have been left in to give you an idea of how I expect you to tackle the problem, but feel free to ignore them if you have your own ideas. On successfully compiling and running the template, you should receive the plaintext given above.

(f) Key Schedule : Given the key, rotate it left by two bytes, then read the first 24 bits of the key, and the last 24 bits of the key, to obtain the 48 bit subkey. Repeat for every round.