CS 465 Cybersecurity Principles and Practice
Programming Assignment #1

Assigned: Thursday, January 24, 2019
Deadline for electronic submission: 12:59 pm on Wednesday, February 6, 2019
Deadline for hard copy: at beginning of the class on Thursday, February 7, 2019
No assignments will be accepted after 11:59 pm on February 11, 2019

Brief description
Write a program which implements a secret key encryption system based on Data Encryption Standard (DES). Use the program to

a) encrypt a given plain text with DES
b) decrypt the encrypted text with DES.

To develop your DES implementation use the detailed specification of DES given in the Federal Information Processing Standards Publication published by the National Institute of Standards and Technology http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

Text preprocessing
Assume that the plain text consists of printable ASCII characters (20 – 7E hex, that is, 32 – 126 decimal). Only digits ('0' – '9') and letters ('A' – 'Z', 'a' - 'z') are encrypted, that is, all other characters should be dropped from the text prior to the encryption. (Note that spaces must also be removed). The resulting ASCII text, which consists only of digits and letters, is then converted into binary data by representing each character with its 8 bits long ASCII binary representation (e.g., 'A' = '0100 0001').

After you have converted the text into binary, as required by DES, divide the input into blocks of 64 bits of data and encrypt each block. If the last block is less than 64 bits long, add 0s until it has 64 bits.

Example of plain text: Today is Tuesday.
Plain text with only digits and letters: TodayisTuesday
Binary input to DES (Two blocks of 64 bits. The last 16 bits given in red are added 0s.):
01010100
01101111
01100100
01100001
01111001
01101001
01110011
01010100

01110101
01100101
01110011
01100100
01100001
01111001
00000000
00000000

Generation of key schedule
To allow testing of your program, assume that the key is generated by converting an exactly 8 ASCII characters long key into 56 bits long binary key by using the 7 bits long binary representation of each ASCII character. Bits 8, 16, … , 64 of the key (the blue underlined bits below), which are not used by the algorithm, are used for error detection and assure that each byte is of odd parity (i.e., there is an odd number of 1s in each 8-bit byte).

Example key: password
11100000
11000010
11100110
11100110
11101111
11011111
11100101
11001000

Recall that K_n ($1 \leq n \leq 16$) is the block of 48 bits in the algorithm. Follow the key schedule generation as specified at http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf (pages 18-21).

Input to your program
The input of the program is given as an ASCII text file which contains the plain text. The 8 character long key can be different for each run of your program, that is, it is an input parameter to your program entered by the user (The key "password" above is only an example. Do not hardcode the key in your program; it is an input parameter, not a fixed value.) The output file(s) contains the encrypted text in a binary format, given as above in 64 bits blocks.

Output of your program
To aid in testing and debugging (and in grading), your implementation should output the results of each cycle to a text file, with an indication of what each result is from. See the test case for the format of the output of your program.

Implementation language
The following compilers / interpreters available on LCSEE UNIX server shell are allowed for implementation:

- gcc (Ubuntu 5.4.0-6ubuntu1~16.04.5) 5.4.0 20160609
- Java 8 (openjdk 1.8.0_151)
- Python 2.7.12 & 3.5.2

The program must compile and run on this UNIX machine.

What to submit?

1. **Submit the program files electronically on eCampus by the due date.**
   - A comment block at the beginning of the file(s) must contain the same information provided in the cover and status sheets, described in 2.1 and 2.2.
   - In addition, include a README.TXT file which describes exactly how the program is compiled on the LCSEE UNIX server shell. If you use a Makefile for compilation, include it too.
   - Zip in a single file all program files, data files, and the README.TXT file and submit the zipped file in eCampus.

2. **Submit the following paper copies:**
   - 2.1. **A cover sheet** stating student's name, computer LCSEE login-id, WVU student ID number, the programming assignment number, and the date.
   - 2.2. **A status sheet** which states if your program does not compile or, for working programs, specifies clearly any aspect that does not work properly. This information must be accurate and up to date.
   - 2.3. **Listing of your code** (using small font, e.g. 10 point). Please make sure that your code is commented (inline comments) to explain what parts of the program do.
   - 2.4. **Documentation** must include
     - 2.4.1. Description of the programming environment (language, system, all the assumptions such as libraries, system resources, etc.)
     - 2.4.2. Detailed description which will allow the reader to understand your approach and algorithms used.

Grading

Programs that cannot be compiled will receive at most 40% of the assignment grade, assuming significant amount of functionality has been implemented and adequate documentation was provided. Only well written, nicely structured, well documented programs, which produce correct output will receive full credit.

Academic Honesty

The work submitted for the programming assignment must be your own work. You are not allowed to use any publicly available programs nor programs written by your peers. For the detailed policy of West Virginia University regarding the definitions of acts considered to fall under academic dishonesty and penalties for academic dishonesty, please see the West Virginia University Academic Standards Policy (http://catalog.wvu.edu/undergraduate/coursecreditstermsclassification).