

CSCI 284/162 – Cryptography – Spring 2009
George Washington University

Homework 3

due 27 February ON BLACKBOARD, 6 pm, **on Blackboard**

Policy on collaboration: All examinations, papers, and other graded work products and assignments are to be completed in conformance with The George Washington University Code of Academic Integrity. You may *not* discuss HWs among yourselves. Each student is expected to work independently on his or her own HW; you may not collaborate with others, you may not copy one another's assignments, even in part. You may only refer to your class notes, the text book, and to class material such as slides, notes and links provided on the course website. You may not refer to any other material while working on the homework.

All code must run on your hobbes account. The TA will make no attempt to debug your code, or determine why it does not run. You will be graded on the correctness of your output, and the quality of your code: efficiency and documentation. There will be no exceptions.

Under no circumstances may code be copied from anywhere: classmates, the web, any other source

Any violations will be treated as violations of the Code of Academic Integrity.

Submit all HW in Blackboard by 6 pm on due date. Name your files:

CS284_HW2_LASTNAME_FIRSTNAME.rar or .zip or

CS162_HW2_LASTNAME_FIRSTNAME.rar or .zip

Part A: SPN: Implement an SPN with 4 rows of 4 S-boxes each, each S-box takes as input 4 bits. Use the **same code** for the encryption and the decryption, changing only the variables used by your modules.

Your program takes as input, from a file input_spn.txt, such as the one available off the course website:

- a key K of length 10 bytes, 2 bytes being used for each of five XORs (there is no key schedule as the key bits used are distinct each time)
- the S-box function f as a byte-array of length 16; the first four bits of each byte are 0
- the permutation function σ as a byte-array of length 16; the first four bits of each byte are 0
- m , an integer multiple of 16; m is the length of input in bits
- an input x of length m bits; as a byte array of length $\frac{m}{8}$
- an integer a of value 0 indicating encryption, and value 1 indicating decryption.

Your program produces the following output, in a file named output_spn.txt, such as the one avail-

able off the course website:

- the input key K of length 10 bytes
- the input S-box function f as a byte-array of length 16; the first four bits of each byte are 0
- the input permutation function σ as a byte-array of length 16; the first four bits of each byte are 0
- the value of m , an integer multiple of 16; m is the length of input in bits
- an encrypted or decrypted output of length m bits; as a byte array of length $\frac{m}{8}$; $E_K(x)$ if $a = 0$, and $D_K(x)$ if $a = 1$
- an integer representing \bar{a}

Thus, if output_spn.txt is fed into the code, it should produce input_spn.txt and vice versa.

Part B: Feistel Cipher:

a. Implement a single block of a Feistel cipher assuming an input, from a file named input_feistel.txt, such as the one available off the course website:

- n , the size of each half-block;
- X , the input string of $2n$ bits to be encrypted/decrypted;
- K , a key of length n for the encryption
- an integer a of value 0 indicating encryption, and value 1 indicating decryption.

Your program produces the following output, in a file named output_feistel.txt, such as the one available off the course website:

- n , the size of each half-block;
- Y , the encrypted/decrypted output string;
- K , the encryption key
- an integer representing \bar{a}

Assume the function f is as follows: $f(X_r, K) = X_r \times K \text{ mod } (2^n)$ (with X and K treated as numbers $\text{mod } 2^n$).

Use the **same code** for the encryption and the decryption, changing only the variables used by your modules.

b. Assume the following key schedule: $K_1 = \text{key}$, $K_2 = \sigma(K)$, where σ is a cyclic shift one bit to the right. Implement an encryption module taking the same inputs as the individual Feistel block above, consisting of 2 rounds. The output has the same format as the output in 2a above, this file may be named output2_feistel.txt.

You are expected to write the Feistel Cipher and SPN code yourself, and not to get it from a library.

Submit code for both parts A and B. Your code will be tested by running it on input files like those available off the course website, and checking if the output is correct.