HW2: Length-preserving encryption using Feistel

Due Friday February 24, 11:59pm

We have learned how block ciphers work and how to use them to encrypt messages which are longer than the allowed block size. However, in these encryption schemes, if the message is smaller than the block size, the output ciphertext is normally larger than the input plaintext. In some situations length-extending encryption is problematic, and we want an encryption scheme that outputs a ciphertext of the same length as that of the input plaintext.

In this assignment you have to build an encryption scheme that encrypts a 48-bit message into a ciphertext of length 48 bits. To do so we can rely on the Feistel constructions discussed in class. Implement your cipher as a balanced Feistel network that uses AES as the round function. Following the suggestions underlying the FFX standard, use 10 rounds. Here are the portions of the problem in more detail:

Part 1 (25%): Implement a Feistel cipher which works as follows. Let message m = L_0||R_0, (where '||' represents string concatenation). Let the output of the i-th round of the cipher be L_i||R_i, then output of the (i+1)-th round should be L_(i+1)=R_i, R_(i+1)=L_i $\oplus$ F_K(R_i). Here F_K(.) is the round function built using Advanced Encryption Standard (AES).

Part 2 (15%): Explain in a clearly marked comment within your code why 1, 2, or 3 rounds of Feistel are not enough to provide security by detailing attacks that evidence why the resulting cipher would not behave like a random permutation. These attacks can only abuse the structure of the Feistel, in particular you must assume that AES is secure (behaving as a random permutation). More points will be given for attacks against more rounds.

Part 3 (30%): Using the Feistel round, implement a length preserving encryption scheme for 48-bit messages. Use 10 rounds of Feistel, which is a typical suggested number. (See, e.g., the FFX specification http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffx/ffx-spec2.pdf You do not need to implement FFX, this is just for reference.)

Part 4 (30%): Write detailed unit tests for all functions of your code.

Part 5 (Extra Credit, 10%): Build a second Feistel network that works for credit-card numbers, which are sixteen decimal digits. Your round function should probably use addition mod 10^8 for encryption and subtraction mod 10^8 for decryption. Put the code in a separate class (perhaps called CreditCardFeistel) in MyFeistel.py.

Deliverables: Implement your version in the two Python files, MyFeistel.py and test_MyFeistel.py, available on Github. The test file should be for unit tests and be compatible with pytest. You should not modify the interface (encrypt and decrypt functions) to the MyFeistel and LengthPreservingCipher classes, as during grading we will setup our own test harness to see if it works as expected.

Turn into CMS the two Python source files (.py) with your modifications.