Homework 2

CSC 152 – Cryptography

If anything in this assignment does not make sense, please ask for help.

Quiz: We will have a closed-note 20-30 minute quiz on this homework in class Mon Feb 18.

Due: You should consider the due date to be when you take the quiz because the quiz may cover anything related to this homework, including programming topics. However, anything submitted before grading occurs will be considered on-time.

Non-submitted work:

Read: Go to `http://www.crypto-textbook.com` and under "Sample Chapters" read Chapter 4.

Written Problems:

There are three steps to follow for the written problems. Step 1: Do them as if they were homework assigned to be graded (ie, take them seriously and try to do a good job). Step 2: Self-assess your work by comparing your solutions to the solutions provided by me. Step 3: Revise your original attempts as little as possible to make them correct. Submit both your original attempt and your revision as separate files.

Submit two files to DBInbox following the procedure documented on Piazza. The first file should be named exactly **hw2.pdf** and should be your homework solutions before looking at my solutions. The second file should be named exactly **hw2revised.pdf** and should be your homework solutions after looking at my solutions and revising your answers.

NOTE: If you wish to handwrite your solutions you may, but only if your handwriting is easy to read and the file you submit is less than 1MB in size.

1) Let's say that the key used with AES-128 is `0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F`. Compute the first two round keys used by AES-128 in this case (ie, compute k_0 and k_1 in Fig 4.2 which is also $W[0]$ through $W[7]$ in the Fig 4.5).

2) Using the k_0 and k_1 computed in Problem 1, what is the value of the evolving AES block after "round 1" in Fig 4.2 if initially the AES block ("plaintext x" in Fig 4.2) is `0xFF, 0xFE, 0xFD, 0xFC, 0xFB, 0xFA, 0xF9, 0xF8, 0xF7, 0xF6, 0xF5, 0xF4, 0xF3, 0xF2, 0xF1, 0xF0`.

3) GF(16) is defined like GF(256) except the polynomials all have degree less than 4 and the modulus is $x^4 + x + 1$. Calculate the following, each digit representing a field element in hexadecimal. (a) $5 + F$. (b) $5 - F$. (c) $5 \cdot F$. (d) $5/F$. Note that $5 - F$ is shorthand for $5 + (-F)$ where $-F$ is F's additive inverse, and $5/F$ is shorthand for $5 \cdot (F^{-1})$ where F^{-1} is F's multiplicative inverse.

4) Let's define $f : \{0,1\}^{128} \rightarrow \{0,1\}^{128}$ as the sequence of AES operations Byte Substitution, Shift Rows, Mix Columns. Find a pair of inputs to f that differ in a single bit whose outputs have a low hamming-weight symmetric difference (ie, $f(x) \oplus f(x')$ has a low number of 1 bits). Explain how you found your answer. This gives you a notion of how much diffusion is in a single round. Over multiple rounds this diffusion accelerates to affect all bits.

Programming:

Read: C program requirements at `http://krovetz.net/152/programs.html`.

A) Write a C function `P152` with the following header.

```
void P152(unsigned char dst[64], unsigned char src[64])
```

The program should compute the same thing as the following pseudocode description.

```
// All constants and variables are uint32_t

mix(a,b,c,d):
    a = a + b
    d = (a xor d) >>> 8
    c = c + d
    b = (b xor c) >>> 11
    a = a + b
    d = (a xor d) >>> 16
    c = c + d
    b = (b xor c) >>> 31
    return (a,b,c,d)

round(t0, ..., t15):
    (t0, t4, t8, t12) = mix(t0, t4, t8, t12)
    (t1, t5, t9, t13) = mix(t1, t5, t9, t13)
    (t2, t6, t10, t14) = mix(t2, t6, t10, t14)
    (t3, t7, t11, t15) = mix(t3, t7, t11, t15)
    (t0, t5, t10, t15) = mix(t0, t5, t10, t15)
    (t1, t6, t11, t12) = mix(t1, t6, t11, t12)
    (t2, t7, t8, t13) = mix(t2, t7, t8, t13)
    (t3, t4, t9, t14) = mix(t3, t4, t9, t14)
    return (t0, ..., t15)

P152(unsigned char dst[64], unsigned char src[64]):
    (c0, ..., c15) = round(round(0, ..., 15))    // Can be precomputed
    (x0, ..., x15) = read src little-endian in 4-byte chunks
    (x0, ..., x15) = (x0, ..., x15) xor (c0, ..., c15)
    do 6 times:
        (x0, ..., x15) = round(x0, ..., x15)
    dst = (x0, ..., x15) written little-endian in 4-byte chunks
```

The best way to validate something cryptographic is to have multiple independent implementations agree on inputs and outputs. So, I suggest that once you think you've done a good job, you post to Piazza some inputs and outputs. Once a critical mass agrees, then it's probably correct.

Submit your function via DBInbox in a file named exactly `hw2_P152.c`.