Graded Homework

CSC 152 – Cryptography

Due: This work should be completed before you take your module quiz.

Graded programming work has technical and non-collaboration requirements. Read "Program requirements" at the course webpage (`http://krovetz.net/152`) before doing this assignment.

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

Programming:

A) Write a C function `perm152` with the following header and using SSE registers for all of the update operations.

```
void perm152(unsigned char *in, unsigned char *out)  // each an array of 64 bytes
```

This function reads 64 bytes from `in` and writes 64 bytes to `out` as specified by the following pseudocode.

```
update(w,x,y,z):
    w = w + x; z = z ^ w; z = rotl(z, 16);
    y = y + z; x = x ^ y; x = rotl(x, 12);
    w = w + x; z = z ^ w; z = rotl(z, 8);
    y = y + z; x = x ^ y; x = rotl(x, 7);

perm152(unsigned char in[64], unsigned char out[64]):
    read from in to 4 SSE registers (a0 a1 a2 a3), (a4 a5 a6 a7), ..., (a12 a13 a14 a15)
    10 times do:
        do the SSE equivalent for the following
        update(a[0], a[4], a[8], a[12])
        update(a[1], a[5], a[9], a[13])
        update(a[2], a[6], a[10], a[14])
        update(a[3], a[7], a[11], a[15])
        update(a[0], a[5], a[10], a[15])
        update(a[1], a[6], a[11], a[12])
        update(a[2], a[7], a[8], a[13])
        update(a[3], a[4], a[9], a[14])
    write the 4 SSE registers to out
```

We will define all `uint32_t` memory reads and writes to be little-endian, but since this is how our computers do things natively, you do not have to write any extra code to achieve it.

Submit one file with only one non-static function `perm152` via Fileinbox in a file named **hw3_perm152.c**. (Your file may have as many static functions as you want.)

B) Write a C function `perm152ctr` with the following header.

```
void perm152ctr(unsigned char *in,    // Input buffer
                unsigned char *out,   // Output buffer
                int nbytes,           // Number of bytes to process
                unsigned char *block, // A 64-byte buffer holding IV+CTR
                unsigned char *key,   // Key to use. 16-32 bytes recommended
                int kbytes)           // Number of key bytes <= 64 to use.
```

This function should perform CTR mode using perm152 as a block cipher. If we let `paddedKey` be the key followed by as many zero bytes as are needed to make it 64-bytes long, then the block cipher to use is defined as `perm152-BC(blk,key) = perm152(blk xor paddedKey) xor paddedKey`.

The block passed in will be filled with the IV followed by the counter. That block should be used to produce the first block of keystream and then incremented for each subsequent block of needed keystream. Code for incrementing the counter is as follows.

```
int i = 63;
do {
   block[i] += 1;
   i -= 1;
} while (block[i+1] == 0);
```

Note that this simulates how the counter would be incremented if it were being incremented on a big-endian computer (ie, counter read into a register without byte-reversal, incremented, and then written back to memory without byte reversal). This is traditional for CTR mode for some historical reason.

Submit one file with exactly one non-static function `perm152ctr` via Fileinbox in a file named **hw3_perm152ctr.c**. (Your file may have as many static functions as you want.)

Watch Piazza for help with this problem. I will post it early next week.