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.

```
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]):
    let a be an array of 16 uint32_t variables
    copy 64 bytes from in to a
    10 times do:
        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])
    copy 64 bytes from a 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 **hw2_perm152.c**. (Your file may have as many static functions as you want.) The best way to test your function is via crowdsourcing. People should start a thread on Piazza asking and posting inputs, outputs and maybe intermediate values. Once a lot of people agree on the results of some well-chosen inputs, the implementation is probably correct.

B) Write a C function `perm152inverse` with the following header.

```
void perm152inverse(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` and should be the inverse function of `perm152`. In other words, if `a`, `b`, and `c` are all 64-byte buffers, then `perm152(a,b)` followed by `perm152inverse(b,c)` should always result in buffers `a` and `c` having identical contents.

Submit one file with exactly one non-static function `perm152inverse` via Fileinbox in a file named **hw2_perm152inverse.c**. (Your file may have as many static functions as you want.)