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.

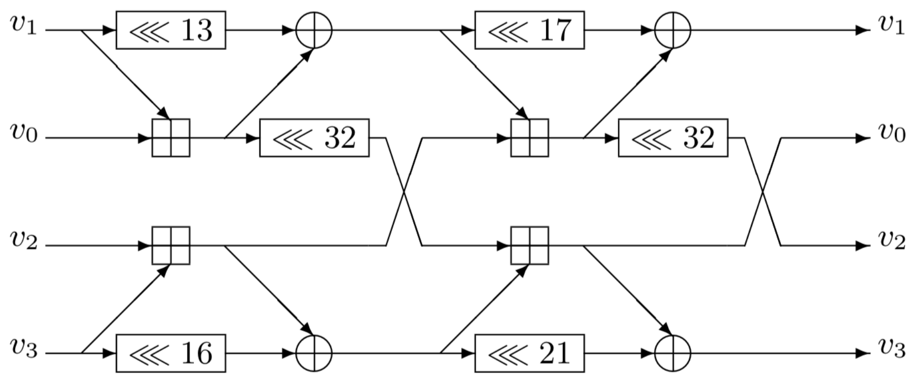
If anything in this assignment does not make sense, please ask for help.

Programming:

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

```
void hash_round(uint64_t *v)      // v[0], v[1], v[2], v[3] are each uint64_t
```

The value `v` is a reference to 32 bytes of memory, which this function intends to mix together. The function should implement the following diagram, where <<< represents left rotation, square plus is addition mod 2^{64} (ie, what you get when you add `uint64_t`'s together) and round plus is exclusive-or.



A twist. The natural memory order of most computers is little-endian, which means that as the bytes are read from or written to memory they are byte-reversed en route. Let's define our function as big-endian instead, which means that we need to manually undo the automatic byte-reversal when we read the data and again when we write it. Here's what I mean in pseudocode.

```
uint64_t v0 = v[0]      // v0 is 8 bytes but opposite byte order of memory
v0 = reverse_bytes(v0)  // v0's bytes are now in the same order as memory
uint64_t v1 = reverse_bytes(v[1]) // Can compose the two steps into a single line
uint64_t v2 = reverse_bytes(v[2])
uint64_t v3 = reverse_bytes(v[3])
... implement the picture ...
v[0] = reverse_bytes(v0)
v[1] = reverse_bytes(v1)
v[2] = reverse_bytes(v2)
v[3] = reverse_bytes(v3)
```

Submit one file with one non-static function `hash_round` via Fileinbox in a file named **hw1_hash_round.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) At `http://krovetz.net/152/module_c/hw1_openssl_encrypt.c` will be soon an example program that encrypts a file. On the command line it takes a key/password and two file names. It then uses the key to encrypt the first file and writes the result to the second file. Write a complete program in C that has the same command line interface as this one, but instead decrypts the first file writing the result to the second file. You may assume that the first file was written using the above encryption program. You may also assume that no errors will occur (this is a bad idea for professional practice, but simplifies the exercise).

Submit one file with one non-static function `main` via Fileinbox in a file named **hw1_openssl_decrypt.c**. (Your file may have as many static functions as you want.) The best way to test your function is to create various test files, run them through the encrypt program, and then run the result through your decrypt program. If the end result matches the original, it's working.

Watch Piazza for extra help that will appear in the Q&A in the next week.