Homework 1

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 Fri Feb 8.

Non-submitted work:

Read: "Homework procedures" found on Piazza and submit your "code.txt" file to DBInbox.

Read: Go to `http://www.crypto-textbook.com` and under "Sample Chapters" read Chapter 1 through Page 20 for a brief introduction to several areas we will study in this class.

Read: as needed, the notes found on Piazza entitled "Permutations" and "C topics useful for cryptography".

Do: Enroll in the Coursera course `https://www.coursera.org/learn/pointers-arrays-recursion`. You can opt for a free/audit version. Review the videos, readings and do the quizzes from Weeks 1, 2 and 3 (ignoring any project-related materials). This is the level of understanding you must have to complete the programs in this class. If you don't understand the Coursera materials well enough to use them in your programs, you should take that class and/or the free version of the edX course `https://www.edx.org/course/c-programming-pointers-and-memory-management` and find time to complete it as soon as possible.

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 **hw1.pdf** and should be your homework solutions before looking at my solutions. The second file should be named exactly **hw1revised.pdf** and should be your homework solutions after looking at my solutions and revising your answers.

1) As you may have seen in CSC 28, if f is a binary relation between sets A and B with $|A|$ ordered pairs in the relation and every element of A occurs exactly once as a first element in an ordered pair, then the relation can be viewed as a function $f : A \to B$.

a) Let $A = \{1,2,3\}$ and $B = \{a,b,c\}$. Is $f = \{(1,a),(2,b)\}$ an invertible function? If so, what is its inverse (given as a set of ordered pairs). If not, explain in one short sentence.

b) Let $A = \{1,2,3\}$ and $B = \{a,b,c\}$. Is $f = \{(1,a),(2,b),(3,b)\}$ an invertible function? If so, what is its inverse (given as a set of ordered pairs). If not, explain in one short sentence.

c) Let $A = \{1,2,3\}$ and $B = \{a,b,c\}$. Is $f = \{(1,a),(2,b),(3,c)\}$ an invertible function? If so, what is its inverse (given as a set of ordered pairs). If not, explain in one short sentence.

2) Let $\mathbb{Z}_n$ be shorthand for the set containing the n smallest non-negative integers (eg, $\mathbb{Z}_3 = \{0,1,2\}$). Is $f : \mathbb{Z}_5 \to \mathbb{Z}_5$ defined as $f(x) = 2x \bmod 5$ an invertible function? If so, what is its inverse (given either as a set of ordered pairs or as a formula). If not, explain in one short sentence. *Note: since this signature is of the form $A \to A$, if it is invertible then it can also be called a permutation or permutation function.*

3) a) How many functions exist with signature $f : \mathbb{Z}_4 \to \mathbb{Z}_5$?

b) Given that a and b are positive integers, how many functions exist with signature $f : \mathbb{Z}_a \to \mathbb{Z}_b$?

c) How many permutation functions exist with signature $f : \mathbb{Z}_4 \to \mathbb{Z}_5$?

d) How many permutation functions exist with signature $f : \mathbb{Z}_4 \to \mathbb{Z}_4$?

e) Given that a and b are positive integers, how many permutation functions exist with signature $f : \mathbb{Z}_a \to \mathbb{Z}_b$?

4) Let `rand(n)` be a library function that evaluates to a random integer in $\mathbb{Z}_n$ each time it is called (like Java's `Random.nextInt(n)`). Write a method called `createRandomFunction` (right here in your written homework) in C or Java that takes a positive integer n as a parameter and returns an array with n elements each uniformly distributed in $\mathbb{Z}_n$. Essentially I'm asking you to write a method that specifies a random function $\mathbb{Z}_n \to \mathbb{Z}_n$ using the table filling method (ie, `a = createRandomFunction(10)` fills `a` with random values and then `a[0]` would tell you what 0 maps to, `a[1]` tells you what 1 maps to, etc.).

5) Do Problem 4 again, but this time name the method `createRandomPermutation` and make the array a permutation (ie, 0 through $n-1$ each appear exactly once). For full credit, make your method run on $O(n)$ time.

6) Let a and b be integers greater than 2, and let $f : \mathbb{Z}_a \to \mathbb{Z}_b$ be a random function. (a) What is $\Pr[f(0) = 0]$? Explain. (b) What is $\Pr[f(1) = 1 \mid f(0) = 0]$? Explain. (c) What is $\Pr[f(1) = 0 \mid f(0) = 0]$? Explain.

Let a and b be integers greater than 2, and let $f : \mathbb{Z}_a \to \mathbb{Z}_b$ be a random permutation function. (d) What is $\Pr[f(0) = 0]$? Explain. (e) What is $\Pr[f(1) = 1 \mid f(0) = 0]$? Explain. (f) What is $\Pr[f(1) = 0 \mid f(0) = 0]$? Explain.

7) A *string* is a concatenation of symbols from an alphabet. For example a string of bits is a concatenation of elements from the alphabet $\{0, 1\}$ and a string of bytes is a concatenation of elements from the alphabet $\{x \mid x \text{ is an 8-bit string}\}$. A String in Java is the concatenation of elements from the type char. (a) Let A be the set of all bytes (ie, all 8-bit strings). How many elements are in A? Explain. (b) Let B be the set of all 8-byte strings (ie, all 64-bit strings). How many elements are in B? Explain.

Programming:

Due: The programs will be first graded sometime after 6am, Sunday, Feb 10.

Read: Program technical requirements at `http://krovetz.net/152/programs.html`.

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

```
    uint32_t sum(unsigned char data[], unsigned int data_bytes)
```

The value `data` is a reference to `data_bytes` bytes of memory. The function should read each consecutive 4 bytes of data big-endian and treat it as a `uint32_t`. Return the sum of all these values (mod 2^{32}). If `data_bytes` is not a multiple of 4, ignore the last `data_bytes` mod 4 bytes of the buffer. Submit your function via DBInbox in a file named `hw1_sum.c`.

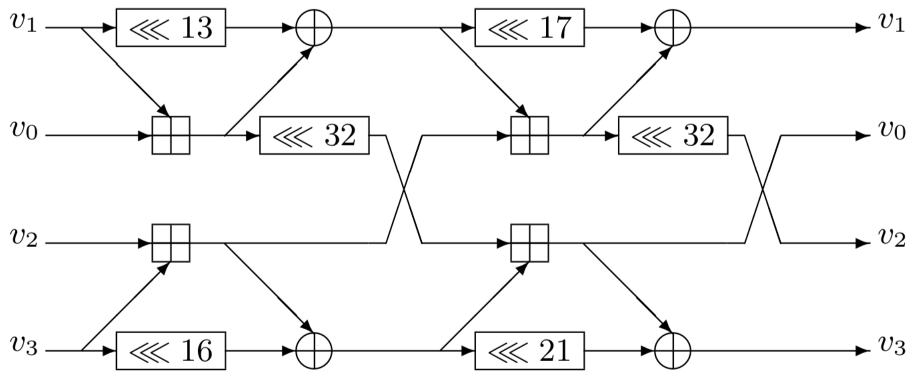
Note: If `p` is a pointer to an unsigned char, you can read 4 bytes big-endian into an uint32_t by following this pseudocode.

```
Read p[0], cast it to uint32_t, shift it left 24 bits,
Read p[1], cast it to uint32_t, shift it left 16 bits,
Read p[2], cast it to uint32_t, shift it left 8 bits,
Read p[3], cast it to uint32_t.
Bitwise-Or these four values.
```

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

```
    void hash_round(uint64_t v[4])
```

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} and round plus is exclusive-or.



Note: If you are using a little-endian CPU (which you almost certainly are), then when you do something like `uint64_t v1 = v[1]` eight bytes are read from memory and placed in a register little-endian, and when you do `v[1] = v1` eight bytes are written to memory from a register little-endian. A lot of cryptography takes little-endian dominance into account and specifies that reads and writes are done little-endian. So, for this problem, let's do the same. The full definition of what you should implement is as follows. But note that the 64-bit reads and writes that your computer does are naturally little-endian, so the `read64LE` and `write64LE` should not be part of your program. They happen for free as part of your array reads and writes.

```
v0 = read64LE(v+0); v1 = read64LE(v+1); v2 = read64LE(v+2); v3 = read64LE(v+3);
Do picture
write64LE(v0, v+0); write64LE(v1, v+1); write64LE(v2, v+2); write64LE(v3, v+3);
```

Submit your function via DBInbox in a file named `hw1_hash_round.c`.

C) We saw in class that one way to write an invertible function is to use a Feistel construction. Here's an example C function that is invertible and uses x^2 as its mixing function.

```
uint32_t perm(uint32_t x) {
    uint16_t hi = (uint16_t)(x >> 16);        // hi initialized from high 16 bits of x
    uint16_t lo = (uint16_t)(x & 0xFFFF);     // lo initialized from low 16 bits of x
    hi = hi ^ (lo * lo);                      // Feistel step: hi = hi xor f(lo)
    lo = lo ^ (hi * hi);                      // Feistel step: lo = lo xor f(hi)
    x = ((uint32_t)hi << 16) | (uint32_t)lo;  // Reform x from hi and lo
    return x;
}
```

Write its inverse as a function with the following header.

```
    uint32_t perm_inverse(uint32_t x)
```

It should be that `perm_inverse(perm(x)) == x` for every `uint32_t y`. Submit your function via DBInbox in a file named `hw1_perm_inverse.c`.