Assignment 3 – Due Wednesday, February 26

Chapters 5,6

1. Select a random 8-bit value (with at least 2 bits set to 1), and show the work required to find the AES S-box mapping for this value. Your final result should agree with the look-up table given in Table 5.2.

2. For this question, you are to work through the first four steps of AES encryption (the initial AddRoundKey followed by the first 3 steps in round 1). Show enough work to demonstrate how you calculated the result of each step.

   (a) Pick a 16 byte plaintext and 16 byte key. You can pick some random values, or write down some pattern, but don't use anything obvious — I don't want to see any two students with the same values! Write down your values in the 4×4 state matrix form used in the book.

   (b) Perform the initial AddRoundKey step on the values from (a).

   (c) Perform SubBytes on the result of part (b).

   (d) Perform ShiftRows on the result of part (c).

   (e) Perform MixColumns on the result of part (d).

3. The latest versions of Intel and AMD processors support new CPU instructions called the "AES-NI" instructions. Do some research on the Internet and look for benchmarks that indicate how fast AES encryptions can be performed with these instructions. There are several ways of expressing this (encryptions per cycle, bytes per cycle, bytes per second, ...), and some assumptions that affect the results — as long as you're clear in expressing your answer, you may make any reasonable assumptions that you'd like. Make sure you cite your sources in your answer!

4. A system to store grades at the University of Secrecy stores grades in a table with each row containing a student name, course name, and encrypted grade (A-F — no pluses or minuses). To encrypt the grade, the system takes the ASCII character for the grade, puts it in the least significant 8 bits of a 128-bit block with the rest of the bits being 0, and encrypts that block using AES with a secret key. Joe Hacker has stolen this table, but all the grades are encrypted, so what now? Joe has also gained limited access to a grade query program which gives actual grades, but only gets to make 5 queries before he is discovered and kicked out of the system.

   (a) How can Joe make the most of his 5 queries? Describe a process so that after 5 queries he knows every student's grade in every class. (*Big Hint:* What does the table look like if Mary Smart makes all A's?)

   (b) Devise a way to correct this problem, so that Joe will only learn the five grades that he is allowed to query and no more. For full credit your solution must not increase the size of the table, so the encrypted grade should still take up one block (128 bits). An alternate solution,

for almost full credit, can double the size of this entry to 256 bits. (*Hint:* Randomness is awesome.)

5. Alice has encrypted an 800,000 byte file using DES in CBC mode, and the ciphertext file contains the IV followed by the actual ciphertext.
   (a) How long is the ciphertext file? (Explain!)
   (b) The next day, Alice wants to recover some data that she knows is in the last 1000 bytes of the file. How can she efficiently recover the last 1000 bytes of plaintext? How many block decryptions does your technique require? It should be significantly smaller than the number of block decryptions required to decrypt the entire file!
6. Page 200, Problem 6.7
7. Page 200, Problem 6.11
8. For the same reason that it is vital that a one-time pad key to be used only once, CTR mode requires that the initial counter value is not reused. Consider an implementation that uses the original C library `rand()` function: it first takes a 16-bit random value as a seed in `srand()`, and then calls `rand()` to generate 128 pseudorandom bits.
   (a) How many different initial counter values are possible using this technique?
   (b) If seeds are truly random, how many times can this technique be used before the probability of a repeated initial seed exceeds 1/2? (*Hint:* Read about the "Birthday attack" in Wikipedia or in the textbook.)
9. (*Extra Credit*) When XOR is used instead of addition in the standard multiplication algorithm, it results in what is called a "carryless multiplication," which is exactly like doing polynomial multiplication with coefficients from GF(2). The following code shows regular integer multiplication on the left, and carryless multiplication on the right (this code works when inputs are 1 byte long).

```
int intmult(int a, int b) {
    int ans = 0;
    while (a != 0) {
        if (a & 1)
            ans += b;
        a >>= 1;
        b <<= 1;
    }
    return ans;
}
```

```
int polymult(int a, int b) {
    int ans = 0;
    while (a != 0) {
        if (a & 1)
            ans ^= b;
        a >>= 1;
        b <<= 1;
    }
    return ans;
}
```

Newer Intel processors have a special instruction that does carryless multiply in one operation. The only thing missing from making this a fast multiplication operation over GF(2^8) is the modular reduction. Can you devise a way to do this quickly so that you can do multiplication over GF(2^8) in 3 instructions (1 can be a table lookup, but you table should be no more than 256 bytes long).