Homework 3

CSC 152/252 – Cryptography
Due: 11:59pm, Friday, October 12

Read the document "Homework procedures" posted in Piazza resources for how to turn in your homework and policies on collaboration. A small sampling of your written homework and most of your programs will be graded over the semester. To minimize the chance that bad luck causes you a bad grade, do every problem to the best of your ability. Show your work. Ask questions if you need help.

What to turn in: Program files `hw3_P52.c` and `hw3_ctr.c` and a single file with your written solutions named `hw3.pdf`. Submit all three to `http://dbinbox.com/cscx52/xxxx` where `xxxx` should be replaced by your four-digit code. Verify submission using the DBInbox link sent to you earlier in the semester. Mis-submitted or mis-named files will receive no credit.

References: Chapter 5 slides from *Understanding Cryptography* and Modes of Operation section from *Serious Cryptography*.

Written Problems:

1) The AES S-box is a permutation, and therefore could be used in the modes of operation we learned (ECB, CBC, CTR, OFB). Use the S-box in each of the modes to encrypt "abc". In modes that need padding use 10^* padding. For modes that need an IV use 01010011. For modes that need a nonce, use 0110. Note that the S-box would never be used this way, I'm just using it as a readily available permutation for practice.

2) Let's say that a ciphertext that was created using a mode-of-operation has a single bit toggled in it's i-th block before decryption. How damaging is it to the decryption? Describe the damage with respect to errors in the resulting plaintext blocks (eg, "plaintext block i has a single bit error", or "all plaintext blocks later than i look random", etc). Do this for each of the four modes ECB, CBC, CTR, OFB.

3) You are given a black box f that has either a standard 52-card deck-of-cards or a 48-card deck-of-cards for the game pinochle. You are allowed to activate f once, upon which a card is chosen at random and you are given the card. In pseudocode, give an algorithm that uses f once and then guesses either "standard" or "pinochle". Evaluate the advantage your algorithm achieves.

In this problem you can maximize your advantage by identifying all of the cards that are more likely with one type of deck and guess that type of deck if any of those cards appear.

4) You are given a black box f that has either a 30-sided die or a 34-sided die inside. Each time you activate f the die is rolled and you are told the resulting value. You are allowed q activations. In pseudocode, give an algorithm that uses f q times and then guesses either "30-sides" or "34-sides". Evaluate the advantage your algorithm achieves as a function of q.

Programming:

P1) Many symmetric cryptography algorithms are designed to exploit parallelism. As an example, consider `P52`. Its inner-loop processes one column of data per iteration and each iteration is independent and identical to the others. This can easily be processed in a SIMD manner. Rewrite your P52 code to use SSE intrinsics processing all four columns at once. Use the same header as your original P52.

```
void P52(unsigned s[12])
```

I suggest you do this incrementally. Replace the smallest amount of your old code with SSE code and test it until it produces the same result as your old code. Then do it again – replace a small amount and test. Keep doing this until all your old code in the inner-loop has been replaced with SSE code. Next replace the diffusion code that mixes Row 0 with SSE code. Finally, eliminate as many memory

reads and writes as you can. Ideally, after you've done so, you'd read `s` into three SSE registers at the beginning of your function and then write the three back to `s` at the very end, keeping `s` in registers for the entire procedure. For maximum speed you could also unroll the outer loop to do four rounds per iteration (but this is not required).

Put your code into a file `hw3_P52.c` and submit it via DBInbox. Your file should include the required function and no `main`, should never print anything unless specified to do so, should be appropriately documented, and should compile without warning or error when compiled using `gcc` or `clang` with compiler options `-std=c99 -Wall -Wextra -pedantic -c`. The only headers your file is allowed to include are the standard ANSI C headers.

P2) Write a counter-mode encryption program. I will soon put starter code on Piazza that you should adapt for this program. Your program should take two command line arguments: the letter 'e' or 'd' indicating encryption or decryption and a password of up to 48 ASCII characters. For encryption, your program should construct a 12-byte nonce in a reliably unique manner by whatever means are available (eg, `/dev/random`, `ctime`, `time`, etc).

The block cipher to use should be what I called P52-BC in class: P52-BC(k, x) = P52$(x \oplus k) \oplus k$. If k is fewer than 48 bytes, it should have zero-bytes appended to its end to make it 48 bytes long before the xors occur.

The blocks fed to P52-BC during CTR mode processing should be 12 bytes of nonce followed by 36 bytes of counter. Note that the counter should be big-endian incremented between blocks. P52 handles the little-endian reading of data from memory, but it is your responsibility to ensure that the value in memory gets big-endian incremented once per block.

Your program should read from standard input until it closes and write to standard output. Ciphertexts should be constructed as 12-bytes of nonce followed by the counter-mode ciphertext, meaning that ciphertexts are exactly 12 bytes longer than plaintexts.

Do not include your P52 implementation in your file. Instead put the following external declaration in your file.

```
void P52(unsigned s[12]);
```

and compile during testing your `hw3_ctr.c` together with your `hw2_P52.c` (ie, `gcc -std=c99 -Wall -Wextra -pedantic hw3_ctr.c hw2_P52.c`). When testing, I will compile your `hw3_ctr.c` with my `hw2_P52.c` which is why you should not submit a P52 implementation.

Put your code into a file `hw3_ctr.c` and submit it via DBInbox. Your file should include your `main`, should never print anything unless specified to do so, should be appropriately documented, and should compile without warning or error when compiled using `gcc` or `clang` with compiler options `-std=c99 -Wall -Wextra -pedantic -c`. The only headers your file is allowed to include are the standard ANSI C headers.