Ungraded Homework

The following problems are designed to help you understand the material and/or prepare you for exam questions. You should work through as much of the material as you can find the time for.

1 Practice with OpenSSL

Study this program: cbc_encrypt.c. For each function call in the program, find its documentation and read it to get a preliminary understanding of what it does. Probably the easiest way to find the documentation is to do an internet search for each function name with the word *man* (eg, "man fopen" will find you the C manual entry for fopen and "man 1.1.1 EVP_MD_CTX_free" will find you the OpenSSL version 1.1.1 documentation for EVP_MD_CTX_free). Create a document and for each function write one or two sentences that explain in your own words what each function does.

Study this program: cbc_decrypt.c. and find how it is different from cbc_encrypt.c. It has only very minor differences. Briefly describe them.

Log onto a computer that has OpenSSL installed and save the two files cbc_encrypt.c and cbc_decrypt.c. Compile each using the commands.

```
gcc -Wall -lcrypto -o cbc_encrypt cbc_encrypt.c
gcc -Wall -lcrypto -o cbc_decrypt cbc_decrypt.c
```

(Note the `-lcrypto` may need to be at the end instead, depends on which computer you're using.) The `-Wall` commands turn on extra warnings. (It's good practice to fix all such warnings when you program.) The `-lcrypto` tells GCC to link your program with OpenSSL's crypto library. And `-o` tells GCC to name the output something other than `a.out`.

Now experiment by encrypting and then decrypting a file to see that it works. Both `cbc_encrypt` and `cbc_decrypt` take two command line arguments: source and destination file names. So, `cbc_encrypt foo bar` will encrypt the file `foo` and put the result into file `bar` and `cbc_decrypt bar foo2` will reverse the process putting the result into file `foo2`. Look at both output files: the encrypted one should look like random gibberish and the decrypted one should match the original. Be careful with your destination file names; if the file already exists it will be overwritten.

2 Problem from a past exam

Write a function `trade` in C that takes a buffer as input and makes each byte at an even index trade places with the byte just after it. If there are an odd number of bytes in the buffer, the last byte shouldn't move. For example if the buffer was 01020304050607 (7 bytes in hex) before the call, it should be 02010403060507 after.

```
// Trades bytes between even and odd indices.
void trade(void *buf, int buflen) {

}
```

3 Another C problem

Write a function `any_dup_ints` in C that takes a buffer, treats it as an array of int and returns a non-zero int value if any pair of ints in the array are equal, and 0 otherwise. If the buffer is not a multiple of sizeof(int) bytes long, the trailing bytes should be ignored.

Note: C has no boolean type. Instead, boolean expressions evaluate to an int value and 0 is interpreted as false. Every non-zero value is interpreted as true in a boolean expression.