CIS 4360- Homework 1, part 1[*]

Due 11:59pm on September 9th, 2019

Prof. Kevin Butler

1 Written questions {20 points}

(a) {10 points} Show with a counterexample that the Substitution Cipher doesn't achieve perfect secrecy.

(b) {10 points} Consider the following modification to one-time pad (OTP) encryption. Rather than share a single one-time pad, Alice and Bob have shared knowledge of two pads, P_1 and P_2.

Given a plaintext M, Alice creates the ciphertext $C = M \oplus P_1 \oplus P_2$, where $\oplus$ denotes xor and $|M| = |P_1| = |P_2|$ (i.e., the size of the message and the two pads are all equal). To decrypt, Bob takes the ciphertext and xors it with P_1 and P_2; i.e., $D(C) = C \oplus P_1 \oplus P_2$.

Argue that the above scheme is perfectly secure.

2 A Simple, Unencrypted P2P Instant Messenger {30 points}

To introduce you to network programming, you will build a simple unencrypted instant messenger. The program reads from standard input and sends input messages to another instance of the program running on a different machine; received messages are sent to standard output.

Figure 1 shows a conversation between instances of the IM client running on two machines, alice-HW1 and bob-HW1.

Your IM client should use TCP to send messages between hosts. Your program should run between two machines alice-HW1 and bob-HW1.

Program description. Your program should read from standard input and send all input data to the other instance of your application (running on the other host), via TCP/IP over a network port, which should default to port 9999 unless you provide a different port number at runtime.

[*]Last revised on August 28, 2019.

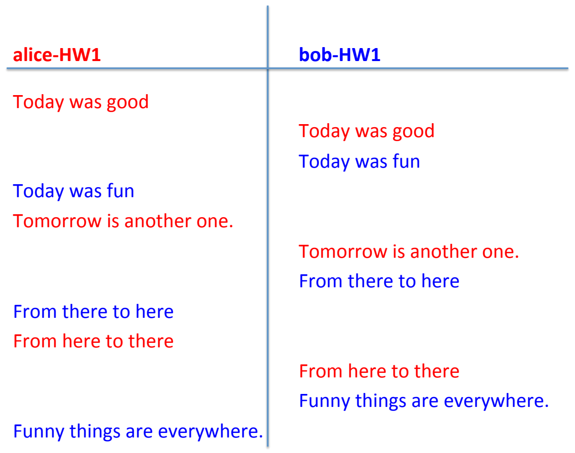

Figure 1: An example conversation between two nodes, as seen by each of the two nodes. Red text is entered by the user at machine alice-HW1 and blue text is entered by the user at bob-HW1.

Here's the tricky bit: Received messages should be *immediately* written to standard output. To do this, you will need to use the *select* call to block and wait for input *either* on standard input or the network socket. For C/C++, you should read the *man* page for the *select* system call. Descriptions of Python's variant of `select` are available online.

(As an alternative to using *select*, you may use multiple threads, although this is far less efficient.)

Your program should be called `UnencryptedIM` (or `UnencryptedIM.ext` where `ext` $\in$ {`.c`, `.cpp`, or `.py`}).

Your program should have the following command-line options:

```
    UnencryptedIM -s <portnum> |-c hostname <portnum>
```

where the `-s` argument indicates that the program should wait for an incoming TCP/IP connection on port 9999 (or optionally the port number specified by `portnum`); the `-c` argument (with its required `hostname` parameter) indicates that the program should connect to the machine `hostname` (over TCP/IP on port 9999, or the port number specified by `portnum`).

As discussed in class, you have access to the Linux machines in the department, or you can run your own Ubuntu 18.04 LTS virtual machine. You may, for example, run "`UnencryptedIM -s 12001`" on lin113-01 to start the server process on port 12001, and then start "`UnencryptedIM -c lin113-01 12001`" on lin113-02 to connect your service on port 12001. Note that the instance with the `-s` option must be started before the other instance.

Additional requirements and hints. Please make sure that your program conforms to the following:

- You may write your program in C, C++, or Python. Please see the teaching staff if you would like to use another programming language. For submissions done in C/C++, we will ignore all submitted executable code and will compile your code from the submitted source files.

- Twenty-four hours after the due date, we will disseminate our solutions to the homework, written in C (possibly C++) and Python. These solutions may be used to complete future parts of HW1.
- You may only use libraries already installed on the Linux machines. For this assignment, this includes socket, select, random, signal, etc. in Python, and the standard libraries in C as well as sys and inet libraries. No crypto libraries are required for this assignment.
- You may not collaborate on this homework. This project should be done individually. You may search the Internet for help, but you may not copy (either via copy-and-paste or manual typing) code from another source. You **may** use code from the textbook, or from the instructor or TAs.
- To aid in automated testing/grading, do not provide a prompt to the user, and only write received messages to standard out. (Text entered into standard input can also be shown; see Figure 1.) We may use automated testing tools to evaluate your solutions, and printing additional messages or characters makes such automation far more difficult.
- Your program should not take in any additional command-line options other than the `-s` or `-c hostname` options described above.
- It is OK if messages are only sent after the user presses [ENTER] after entering a line of text. However, incoming messages should be displayed immediately after they are received by the kernel.
- Your program can terminate either when the user presses CTRL-C, or when end-of-file (EOF) is received. To generate EOF from the terminal, press CTRL-D.
- Hint: To stress test your code, try using your program to copy a file between machines. You should be able to do this by redirecting standard input (at one end) and standard output (at the other).

In part II of this homework (assigned shortly after the due date!), we will add a layer of encryption to our IM applications.

Grading

This programming portion of HW1 is worth 30 points. A non-comprehensive list of deductions is provided in Table 1.

We will award partial credit when possible and appropriate. To maximize opportunities for partial credit, please rigorously comment your code. If we cannot understand what you intended, we cannot award partial credit.

Description	Deduction
Only included executables (no source code; applies to C/C++)	30
Compilation / interpreter errors	20
Compiles, but IMs are neither successfully transmitted nor received	15
Compiles, but IMs are either only transmitted or only received	10
Received messages only appear after user presses [ENTER] (indicates that *select* is used improperly)	5
General instability (e.g., occasional segfaults)	10
Run-time error (e.g., crash) on large input	5
Non-conformant command-line options (hinders automated testing)	5
No compilation instructions provided (applies to C/C++/Java)	5
Includes unnecessary prompts (hinders automated testing)	3

Table 1: Grading rubric. Note that this grading rubric is not intended to be comprehensive.

Submission Instructions

Submit your solution as a single tarball (tar.gz archive) using Canvas. Use the "Homework 1, Part I" assignment.

In the archive, include a single PDF document with your written answers to Questions 1 and 2. **Writeups submitted in Word, ASCII, PowerPoint, Corel, RTF, Pages, and other non-PDF formats will not be accepted.** Consider using LaTeX to format your homework solutions. (For a good primer on LaTeX, see the Not So Short Introduction to LATEX.)

Include in the archive all source code, using the naming conventions described in this assignment. If your program is written in a C/C++, please also provide compilation instructions.

Please post questions (especially requests for clarification) about this homework to Canvas.