Submit this assignment via gradescope: `www.gradescope.ca` An account will be made for you.

Question 1: Threat Models (12 points)

Access to public metros (undergrounds, subways, tubes, light rails, suburban rails, etc.) typically employ a form of access control: would be passengers present valid evidence of the purchase of a ticket (fare, proof of purchase) to a human gatekeeper or a computerized turnstile, or the passenger is expected to carry their ticket on their person while in transit and present if requested during an infrequent inspection. This latter option is also called the "honour system" and it is used by Calgary's C-Train metro system.

Turnstiles and ticket inspections are security mechanisms and they imply a security policy, an adversary, and a threat model.

In 3–4 paragraphs (i.e., written form, not point form), describe the threat model of a metro system. Write it as though you are explaining the concerns of the threat, and proposing solutions to address it. Be sure to include:

- What assets are being protected.
- The security policy of the metro system.
- The adversary model, including the adversary's goal, possible attack vectors, feasibility of the attacks, and how these attacks violate the security policy.
- The three example defenses (human gatekeeper, computerized turnstiles, infrequent inspections) and how they prevent the attack, along with costs and risks to deploy. (You may assume that computerized turnstiles are significantly cheaper than human labour.)

Question 2: Adversarial Modelling (12 points)

Recently in the United States, their equivalent of parliament buildings were unlawfully entered by large numbers of people who appeared to be supporters for the president at the time of the breach for the purpose of a *denial of service* attack on a ceremonial counting of votes. There was also theft and destruction of property, which included theft of papers and laptops from the desks and offices of some parliamentarians.

For context, here is a news article:
`https://www.reuters.com/article/usa-election-cyber/update-3-laptop-stolen-from-pelosis-office-during-storming-of-us-capitol-says-aide-idUSL1N2JJ1ZT`

For four of the adversary type in the categorical schema, describe a situation for such an adversary who may have *pretended* to be a part of the superficial purpose of the attack so as to easily enter parliament with the others, but with the goal of carrying out a different attack. You may use the theft of laptop example or other examples, either real or hypothesized. Speak to the adversary motives and how easily their attack would be to mount—assuming that the hard part of circumventing the police force that would normally have protected the premises was no longer an issue.

Question 3: Block Cipher Modes (12 points)

Suppose you are required to use 256-bit AES in CBC mode but you are not permitted to send an IV. How does this affect decryption? How can you transform the message so that the original message is completely recoverable?

Using C or C++ syntax (whichever you prefer), give an algorithm to perform the encoding/decoding. Both keys and IVs are 32 byte long buffers (i.e., `uint8_t buf[32]`). For C++ syntax assume the message is a std::string type; for C syntax assume the message is stored as two variables: a `char *` pointer and a `size_t` length.

You may assume the following functions exist:

- `get_key()`: returns the key to use
- `get_IV()`: returns a random IV
- `AES_CBC_256_encrypt(message, key, IV)`: returns the encryption of the message (C++)
- `AES_CBC_256_decrypt(message, key, IV)`: returns the decryption of the message (C++)
- `AES_CBC_256_encrypt(message, len, key, IV, outstr*, outlen*)`: sets encryption of the message in outstr and outlen (C)
- `AES_CBC_256_decrypt(message, len, key, IV, outstr*, outlen*)`: sets encryption of the message in outstr and outlen (C)

Question 4: Timing Randomness (12 points)

In this question you'll measure how long it takes to make randomness from true sources. If you have a UNIX-based machine you can do this locally. Otherwise you'll have to use a virtual machine that has been setup for you. You won't be able to do this directly on the university linux environment, but the virtual machine can run inside it. The instructions for that follow the question. The rest of the question assumes you are at a terminal.

If you type `cat /dev/random` some randomness will appear. Hit control-c to cancel, since this virtual device will just keep emitting randomness. Because it is binary, it interferes with the console (you may need to type `reset` if your console becomes wonky or unreadable.) You can avoid that by piping the output through base64 encoding:

```
cat /dev/random | base64
```

This is a standard encoding that uses 64 symbols to represent data; it is like base10 (decimal), base2 (binary); base64 uses 64 symbols: A–Z, a–Z, 0–9, and + and /.

If you cat the random file, and do something like move the mouse, you'll notice it emitting data faster.

Type `time head -c 10 /dev/random > /dev/null`

This simply reads the first 10 characters from /dev/random, suppresses it from appearing as output, and then also times how long that takes. The output line "real XmY.ZZZs" says that the command took X minutes, Y seconds, and ZZZ milliseconds to run.

With this, answer the following questions about how long it takes to generate randomness. To do this, first empty the entropy pool (i.e., by running `cat /dev/random` first until no randomness is available). You will see that you cannot do this on the university servers (it has a large pool) but you quickly run out on a desktop computer or the virtual machine. If you start to move the mouse around, for example, you'll see it start printing out more.

For each of the following, perform the experiment at least 5 times and report each number as well as the sample mean. Bonus: compute the 95% confidence interval given by the Student t-test.

- How long does it take to generate 10 characters of randomness while not interacting with the machine?
- How long does it take to generate 10 characters of randomness while moving the mouse around? For the VM, this will have to be in the GUI of the VM.
- How long does it take to generate 10 characters of randomness while performing some network traffic, such as sending a large file with `scp` or downloading a large file? On the VM, you can scp a file from the host to the virtual machine. State what you did exactly to generate network traffic?
- How long does it take to generate 10 characters of randomness while typing. For the VM this may be most effectively done over ssh.
- Which of these methods was the most effective at generating randomness?

Virtual Machine Instructions

Here are instructions to access the VM. You will likely need this again for a later assignment, so even if you use your personal machine for the last question, it is still useful to test this out now.

The virtual machine is accessible from the undergrad computing environment. If you are connecting from home, connect first to `linux.cpsc.ucalgary.ca` and then to a lab machine like `zone50-wa`. Then start it by typing:

```
/usr/local/tinycore/tinycore -f <snapshot file>
```

The snapshot file is used to store the state of the VM. (There is an option -c parameter that you'll use in a later assignment as it provides the network traffic.)

You can `ssh` into the VM by following the instructions. It should be `ssh tc@localhost -p <VM's ssh port>` and the password should be `CPSC526Pass`. The port is dynamically assigned and printed to the console when it starts. If you are logging in remotely to the lab machines you should use `ssh -X` to allow X-forwarding for the VM's GUI.