ECED4406 Lab 1: Hash Collision Timings

Due Date: October 7th, 2023.

Groups:

Labs should be completed in groups of 2. Some groups of 1 or 3 will be accepted based on the class size.

Lab groups will be configured in Brightspace to allow students to submit labs online.

Marking:

The lab is worth a total of 20 points:

2 pts – Generate lab formatting, editorial, English etc.

18 pts - Answers to questions (detailed below)

Expected Report Format:

A lab report is required. This lab report for this lab should primarily answer the questions in the lab, it does not require extensive background information.

Pre-requisites: Using Google CoLab

There is an introductory video at https://www.youtube.com/watch?v=v5W8Uff4x0Q . You can freely use "raw" Python in this course, but you may find this useful.

You can find a notebook for this lab at this link: https://colab.research.google.com/drive/1eeR-0vuDQnVRxYr7qgPqGCXk6GVMVmw0?usp=sharing

NOTE: You will need to copy this to your own Google account for it to run. Or you can copy the Python code out and run locally.

Calculating Time & for SHA-2 hash Collisions

Objective:

- Visualize how increasing the hash length increases the cost of performing hash collision brute force attacks.

Setup:

You will generate a hash collision between two strings. The two strings will visually look similar:

String 1: "Students B00123123123 and B0089178923 receive an F on this lab."

String 2: "Students B00123123123 and B0089178923 receive an A+ on this lab."

The two strings will generate a hash collision by padding the second string with spaces, tabs, or other non-obvious characters.

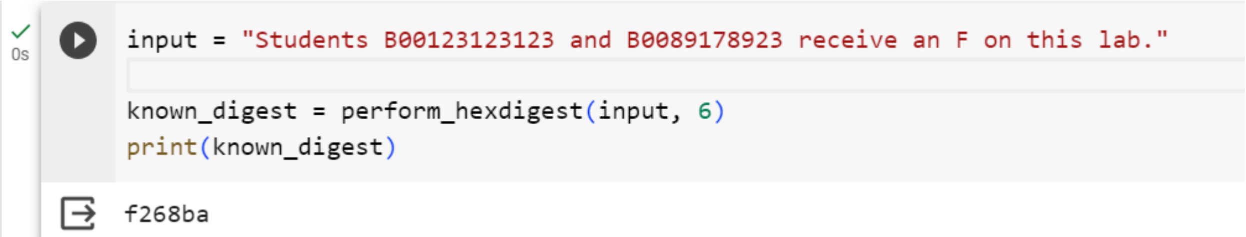
You will use a hash algorithm (such as SHA256), but only take the first 1, 2, 3, .., 6 nibbles of the hash. You will be generating hash collisions against these much shorter strings. For the input string, you can generate a hash value printing only the first 1, 2, 3, 4, 5, 6 nibbles for example of the hash:

Hash Length	Hash Value
1	0xf
2	0xf2
3	0xf26
4	0xf268
5	0xf268b
6	0xf268ba

HINT: It's a common "compression" technique to cut down the full hash to a smaller value depending on the use-case. This demonstration will show why that damages the full cryptographic strength of the hash.

You can generate this one of two ways:

1) Print the full hash value, and simply record only the first character as above.
2) Use the Python function which strips out only the first n characters:



```
input = "Students B00123123123 and B0089178923 receive an F on this lab."

known_digest = perform_hexdigest(input, 6)
print(known_digest)
```

```
f268ba
```

For each of the hash lengths, you will then attempt to "pad" an input string with spaces until you get a matching hash value.

See https://colab.research.google.com/drive/1eeR-0vuDQnVRxYr7qgPqGCXk6GVMVmw0?usp=sharing for reference code for the lab.

Results [18 pts]

1. Your input string, which must be in the following style [1 pt]
`"Students B00123123123 and B0089178923 receive an F on this lab."`

   Replace the B00xxxx numbers with your own banner numbers (adjust as needed for group size).

2. A table showing for hash length of 1, 2, 3, 4, 5, 6 bytes the hash values of the good input string. This looks like the table in the 'Setup' above. [3 pts]

3. A code listing of your code which generates a hash collision. This should be a function which takes as an input (1) the known-good hash, and (2) the string which you need to pad to find a matching hash. The hash collision should be a different string, suggesting the mark you'd instead prefer to receive on the lab. [3 pts]

4. Create a table showing for each hash length, the (a) resulting number of iterations checked before a matching hash is found, and (b) the runtime of each search. Note that depending on your string you may <u>not</u> find hash collisions for the 5 or 6 length hashes in reasonable time (within ~5-10 mins). You can simply report this failure as well, although testing additional bruce force methods beyond just appending spaces may increase your success rate. [6 pts]

5. Briefly describe *why* you can generate hash collisions, and comment on how the length of the search increases with increasing length of the hash. Include a graph showing the progression of brute force iterations with hash length. Again note the 5 or 6 length collisions may not be successful. [5 pts]

~DONE~