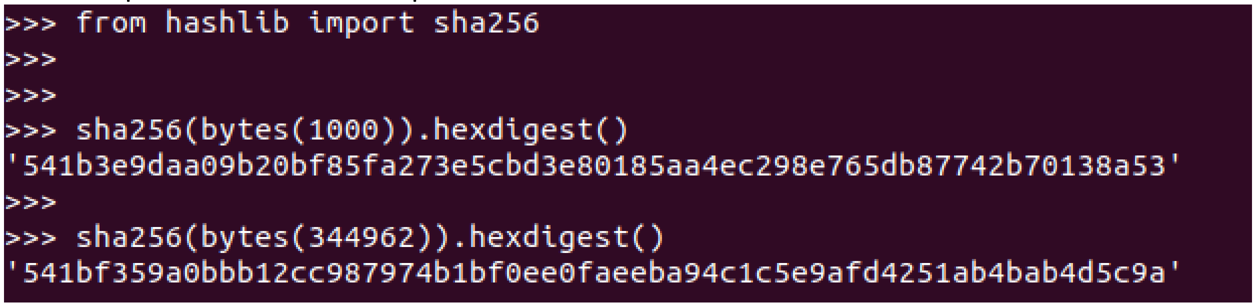
Austin Hunt
CS 6387 – Homework 2 (Crypto Homework) Part 2: Find a Collision
Sept 14, 2022

As we talked about in class, hash functions have an *infinite number of collisions* for arbitrary input sizes, but finding collisions is designed to be prohibitively difficult.
However, finding a partial collision is not impossible.
For example consider the example:



```
>>> from hashlib import sha256
>>>
>>>
>>> sha256(bytes(1000)).hexdigest()
'541b3e9daa09b20bf85fa273e5cbd3e80185aa4ec298e765db87742b70138a53'
>>>
>>> sha256(bytes(344962)).hexdigest()
'541bf359a0bbb12cc987974b1bf0ee0faeeba94c1c5e9afd4251ab4bab4d5c9a'
```

As we see from the first 4 characters in the resulting hashes, these two numbers produce a partial collision with this hash function.

Objective: Find two messages (strings or numbers) whose hexdigests collide in the ***first five characters.***

Total points: 35

Solution
I used multithreading in Python to quickly find partial hash collisions of the first 5 hash characters between various messages, where messages include both:

a) integers ranging from 0 to 1 million
b) random strings of lengths between 1 and 1000

I used the concurrent.futures.ThreadPoolExecutor library to break these large intervals up into chunks of 10,000 and 25 respectively (10,000 chunks for integers, and 25 chunks for random strings of length N). Each chunk is handled by its own thread.

Elapsed time for numeric hash calculations: 4.10352087020874 seconds
Elapsed time for string hash calculations: 0.41069698333740234 seconds
Found 117 partial collisions (including both numbers and random strings)

These 2 numbers/strings when hashed produce a partial collision (shared first 5 hash value characters of c4635): 50018, 160081

I have the code and a lot more partial collision examples provided here:
https://github.com/austinjhunt/vanderbilt-cs6387-cryptography-homework/tree/main/find-hash-collision