CSE 461

INTEGRITY CHECKING AND HASHING

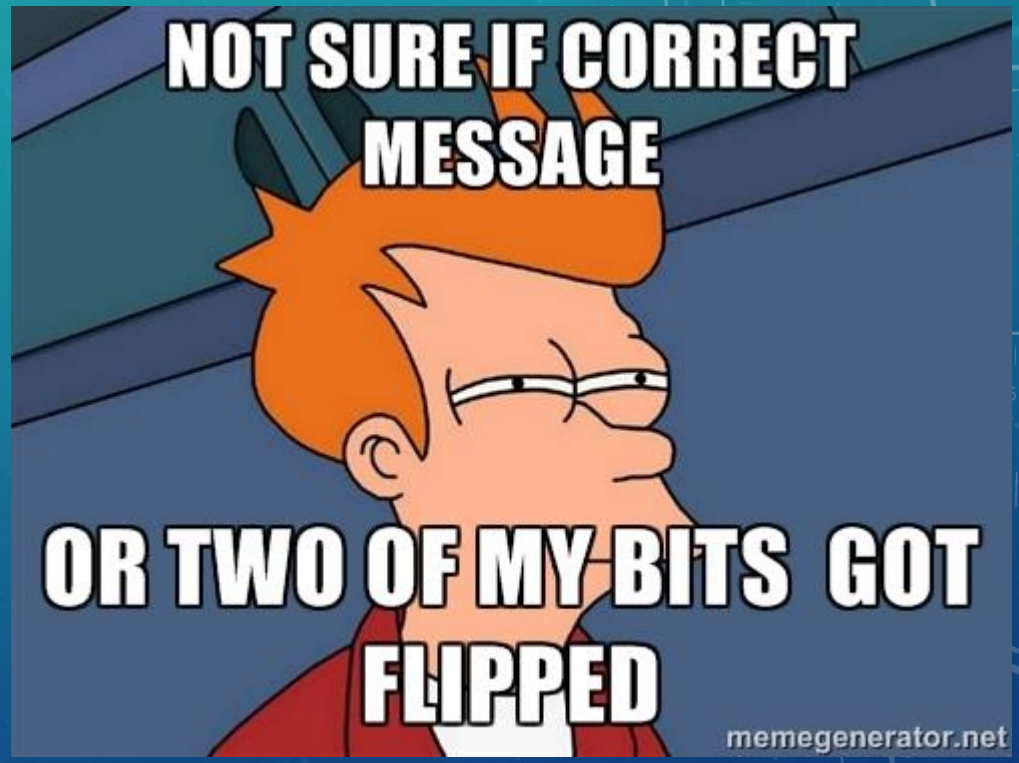
JOKE: TELNET

INTEGRITY

- Want send a file to another party
- Want to make sure the file that arrives is the same data that was sent
- What are some ways we might design a system like this?
  - Send duplicate copies of each bit
  - Send duplicate copies of the entire file
  - Check the results of mathematical functions based on the data

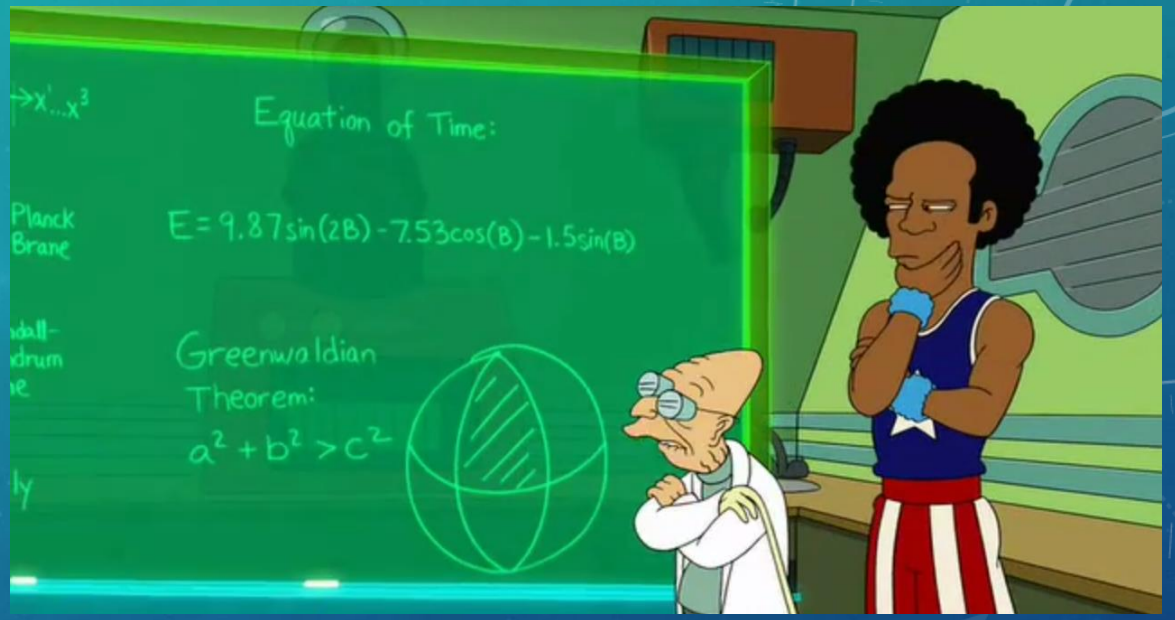
PARITY BITS

- Bits check parity on a set of bits
- Even parity: bits add to 0
- Odd parity: bits add to 1
- Multiple parity bits (on odd bits/ on even bits, etc.) can increase effectiveness
- **Bonus Question** : What parity bit would need to go in the x to achieve even parity? 0010101x
  - 1



CHECKSUMS & CRCS

- Checksums:
  - Adds all words in data as unsigned numbers, allowing to overflow
  - Sum is then compared to check data integrity
  - Many variations
  - One common usage of hashing algorithms
- CRCS:
  - Specific type of checksum that uses polynomial division
- Both are integrity checks using a fixed size of data
- Checksum demo (cksum, md5sum)



HASHES

- Equivalent to checksums, but more general
- Functions to change a large amount of data into a small amount of data
- Example: (whalers.txt)

We're whalers on the moon,

We carry a harpoon.

But there ain't no whales,

So we tell tall-tales,

And sing our whaling tune.

9bc0135a4cf194424c60dbc9faedcaf3

HASHES: USES

- Checking if a file has been modified (checksum)
- Storing data efficiently in tables (hash tables)
- Detecting duplicate files
- Shortening data
- Proving that you know data without that data needing to be stored
    - Passwords

HASHES AND PASSWORDS

- Linux /etc/passwd and /etc/shadow files
- passwd/shadow file demo
- How can we find passwords, if we have a shadow file?
  - Brute force
  - Reverse the hash
    - Cryptographic hashes
  - Rainbow tables
- Brute force password cracking demo

PROTECTING AGAINST ATTACKS

- How can we protect against traditional password attacks? (Assume we're using a cryptographic hash.)
  - Use a deliberately slower hashing algorithm (e.g., Blowfish)
  - Use more secure passwords
  - "Salt" our hashes with extra data
  - Repeated hashing

ANY QUESTIONS?